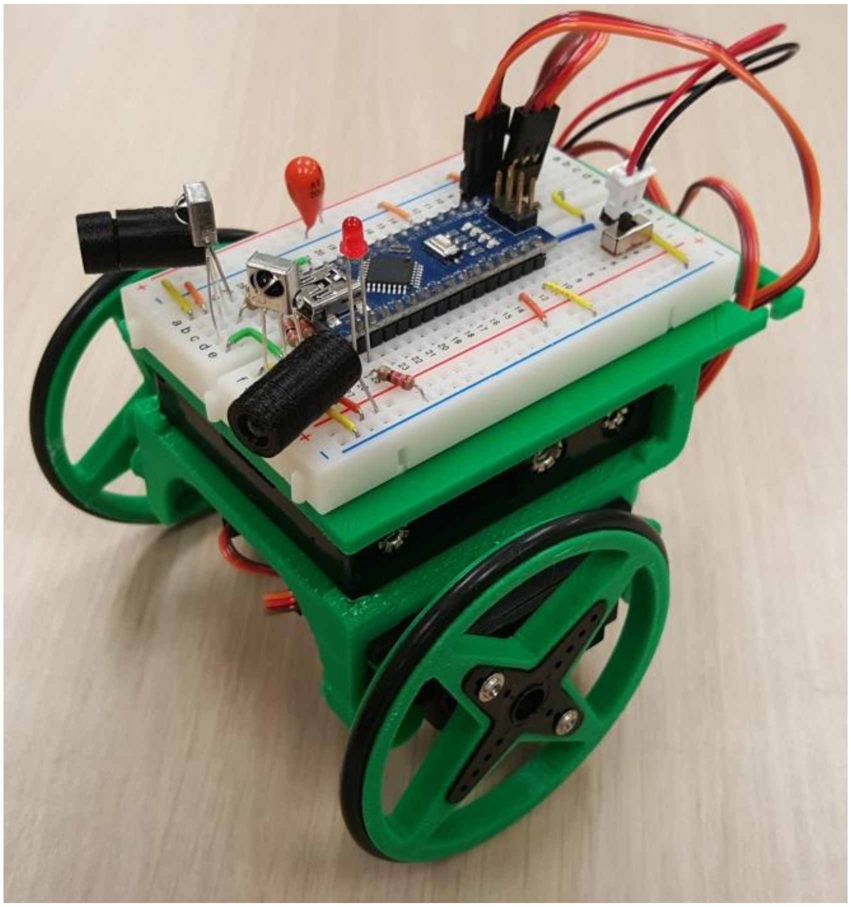


Introduction

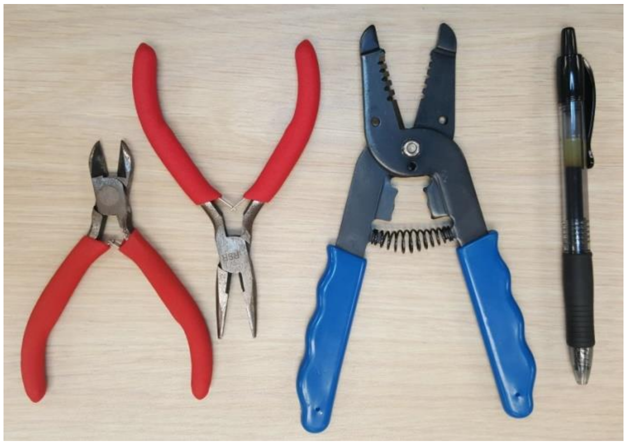
I set out to create this robot kit to give teachers, students, and hobbyists an affordable way to start learning and sharing robotics in their community. Most robotics kits that have the same functions of this kit cost upwards of $150. Most schools, clubs, and maker spaces can afford a few, but not one for every 1 to 2 students. When I took on this project, I wanted to create a kit that people could put together themselves for under $40.

I have included a Bill of Materials (BOM) PDF that has links to different websites to buy all the items needed to build this kit. Prices and availability will change, but hopefully this is a good starting place. I have also included this instruction booklet. The purpose of this instruction booklet is to help you construct and program the ServoBot. There are 4 main parts to this booklet: Part 1: Wiring the circuit board, Part 2: Installing Software and testing the circuit board, Part 3: Constructing the Robot, and Part 4: Programming.

NOTE: If you already have an Arduino UNO, or using a prototyping board is intimidating, you can purchase double sided tape and use the UNO instead.

Part 1. Wiring the Protoboard

Tools Needed:



Small diagonal cutters

Small needle nose pliers

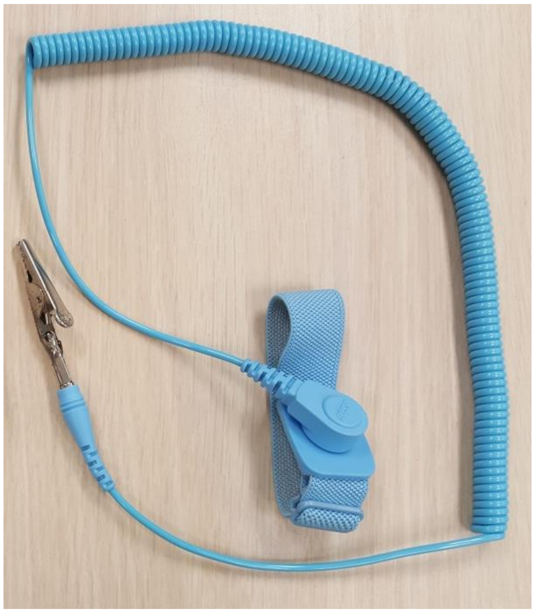
Wire strippers

Pen

You will also need the following safety equipment.

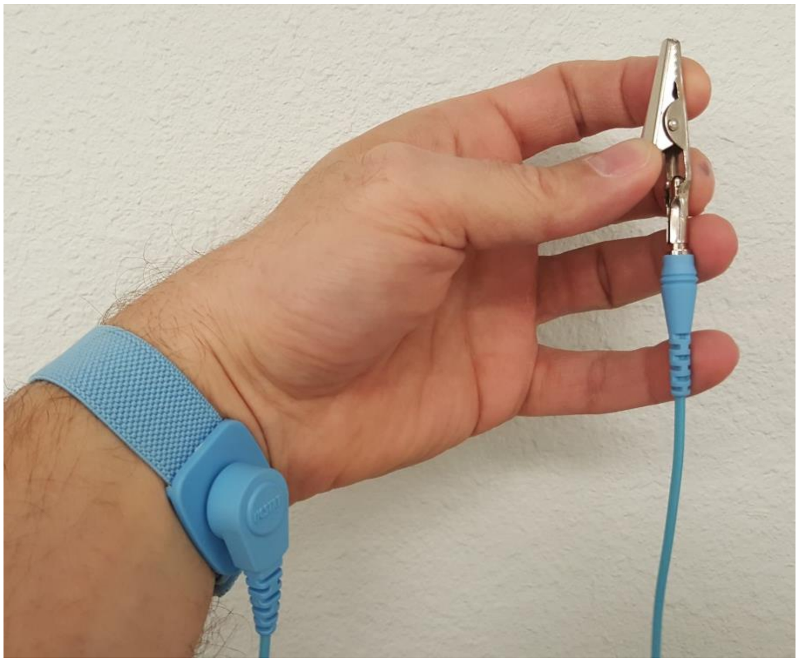
Safety glasses.

NOTE: You will need to wear safety glasses while wiring in case wires fly at your eyes when cutting them. Make sure the glasses cover the sides of your eyes. If you wear glasses, get safety glasses that will fit over your regular glasses comfortably. Instead of safety glasses, you may install side shields on your regular glasses and use them if they fully cover your eyes.



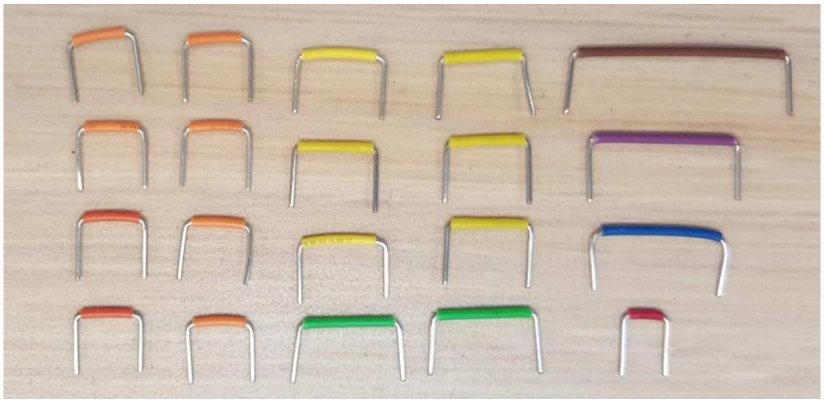
Anti-static wrist strap.

NOTE: You will need the anti-static wrist strap to discharge your body from any static electricity it might be holding so that you do not damage any electronics while you are wiring. When putting it on, make sure the metal pad on the wrist strap is touching bare skin. Attach the alligator clip to ground or at least a large piece of metal.

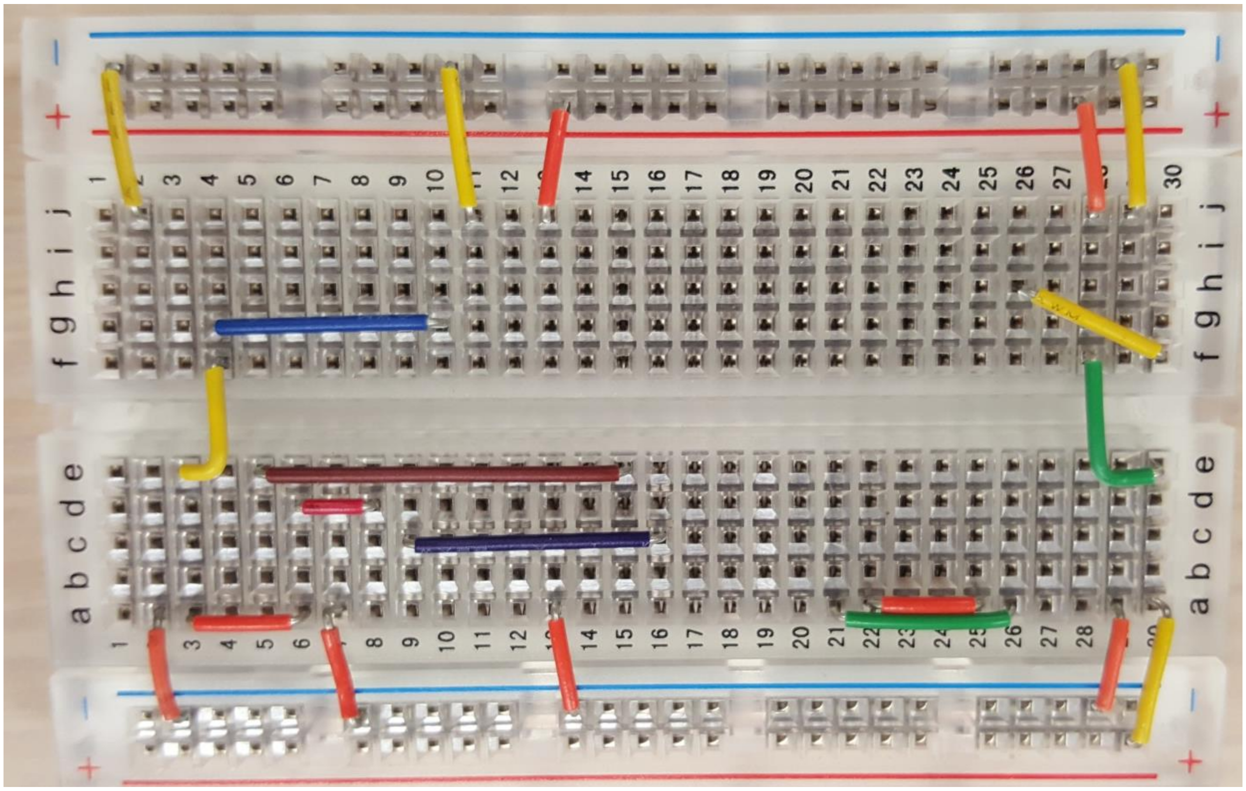


NOTE: Make sure to bind any loose clothing or hair and remove any jewelry.

Materials Needed for Wiring:



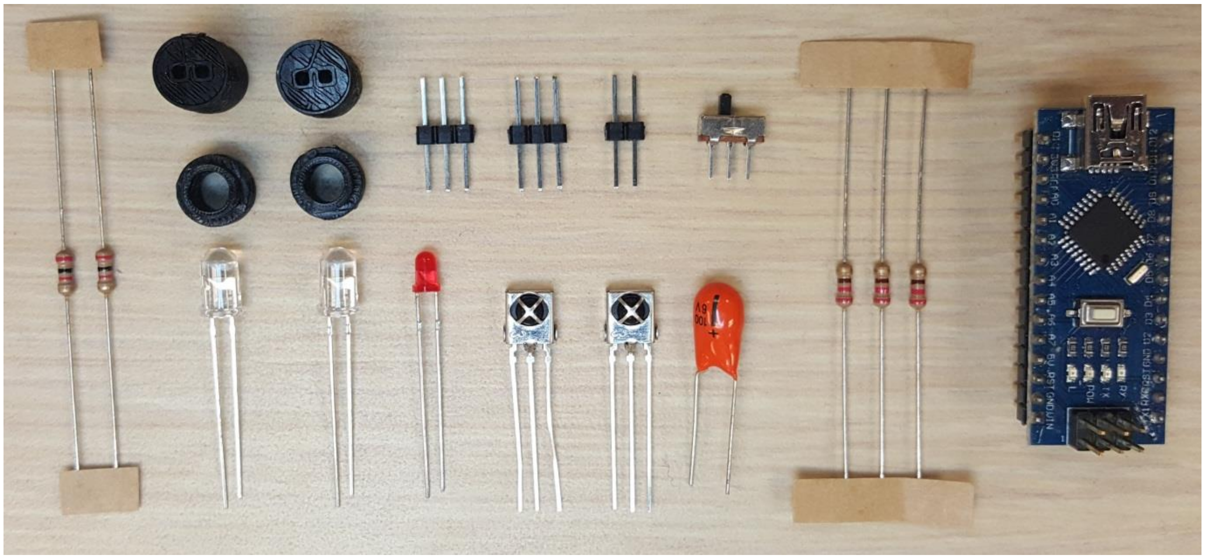
These are the wires needed to wire the circuit below. 8 orange, 6 yellow, 2 green, 1 red, 1 blue, 1 purple, 1 brown. They are found in most proto board wiring kits.



1. Red wire goes from d6 to d8.
2. Orange wire goes from a2 to the Blue Bus Bar next to row 2.
3. Orange wire goes from a3 to a6.
4. Orange wire goes from a7 to the Blue Bus Bar between row 7 and 8.
5. Orange wire goes from a13 to the Blue Bus Bar between row 13 and 14.

6. Orange wire goes from a22 to a25.
7. Orange wire goes from a29 to the Blue Bus Bar between row 28 and 29.
8. Orange wire goes from j13 to the Red Bus Bar between row 13 and 14.
9. Orange wire goes from a28 to the Red Bus Bar between row 27 and 28.
10. Yellow wire goes from a30 to Red Bus Bar between row 29 and 30.
11. Yellow wire goes from j29 to Blue Bus Bar between row 28 and 29.
12. Yellow wire goes from j11 to Blue Bus Bar between row 10 and 11.
13. Yellow wire goes from j2 to Blue Bus Bar between row 1 and 2.
14. Yellow wire goes from e3 to f4.
15. Yellow wire goes from h26 to f30.
16. Green wire goes from e30 to f28.
17. Green wire goes from a21 to a26.
18. Blue wire goes from g4 to g10.
19. Purple wire goes from c9 to c16.
20. Brown wire goes from e5 to e15.

Components Needed for Constructing the Circuit:



IC1: Arduino Nano V3 ATmega328

S1: Single Pole, Double Pole Switch

IR1: Left Infra-Red Receiver

IR2: Right Infra-Red Receiver

R1, R3: 2 kΩ   (Red, Black, Red, Gold)

R2, R4, R5: 220Ω (Red, Red, Brown, Gold)

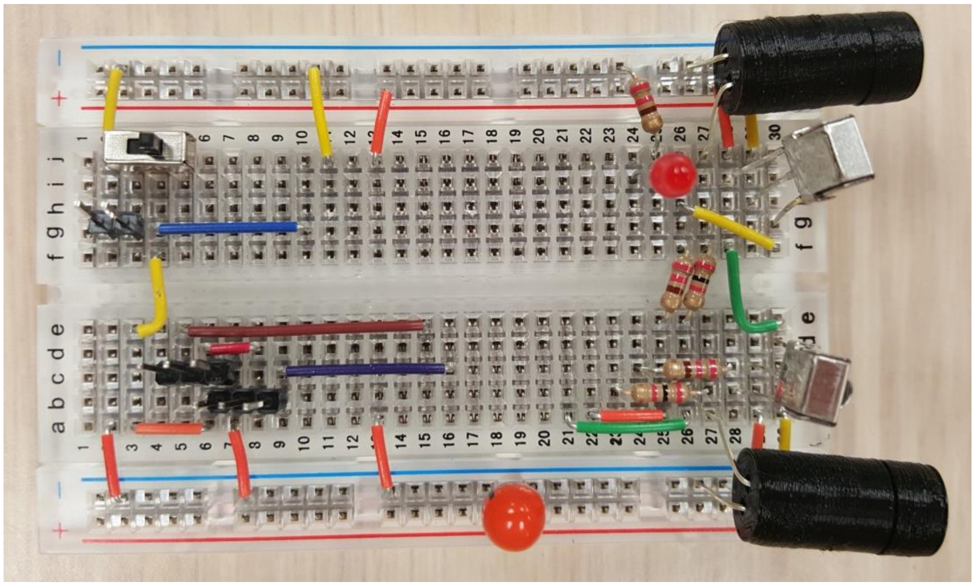
C1: 100μF

L1: Left Infra-Red LED

L2: Right Infra-Red LED

L3: Red LED

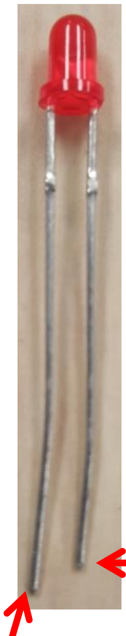
The Electrical schematic for the ServoBot's circuit is on the next page.



1. Capacitor (C1) goes in (Negative Lead) Blue Bus bar and (Positive Lead) Red Bus Bar next to row 18.

   NOTE: Capacitor is polarity sensitive. Either the negative lead or positive lead is labeled. If connected backwards, the Capacitor will explode.

2. Switch (S1) goes in j3, j4, and j5.
3. 2 Pin Header goes in g2 and g3.
4. 3 Pin Header goes in c5, c6, and c7.
5. 3 Pin Header goes in b7, b8, and b9.
6. Resistor R1 (2kΩ) (Red, Black, Red, Gold) goes in e26 and f27.
7. Resistor R2 (220Ω) (Red, Red, Brown, Gold) goes in e25 and f26.
8. Resistor R3 (2kΩ) (Red, Black, Red, Gold) goes in b23 and b27.
9. Resistor R4 (220Ω) (Red, Red, Brown, Gold) goes in c24 and c28.
10. Resistor R5 (220Ω) (Red, Red, Brown, Gold) goes in j25 and the Red Bus Bar close to row 24
11. Red LED's Anode goes in i24, and the cathode goes in i25.



Flat side is the Cathode.

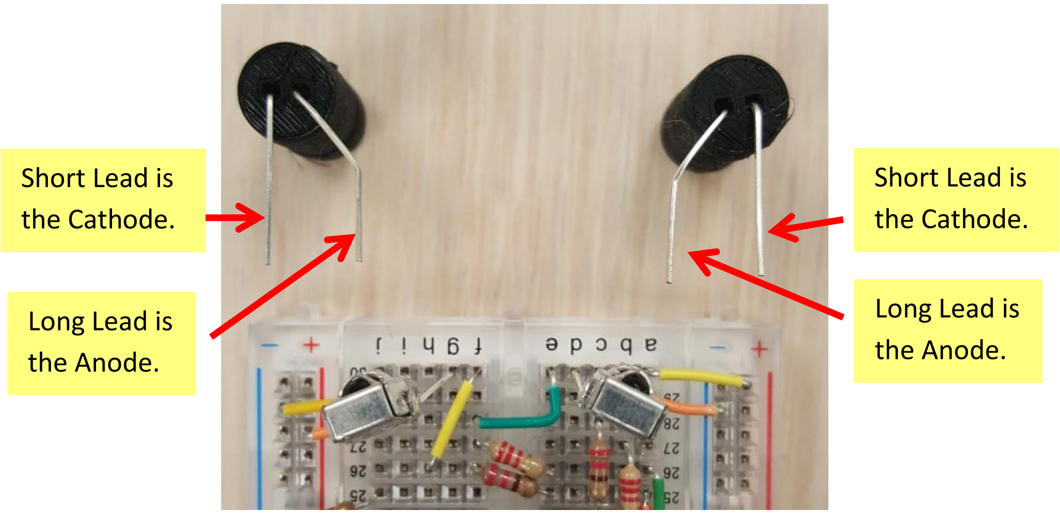
Short Lead is the Cathode.

Long Lead is the Anode.

NOTE: Bend the short lead (cathode) straight down and the long lead (Anode) at an angle, then bend straight half way down with needle nose pliers. Also, make sure to mirror each IR LED. Make them look like the picture below.

12. Right IR LED’s anode goes in a27 and the cathode goes in the Blue Bus Bar between rows 26 and 27.

13. Left IR LED’s anode goes in j27 and the cathode goes in the Blue Bus Bar between rows 26 and 27.



Short Lead is the Cathode.

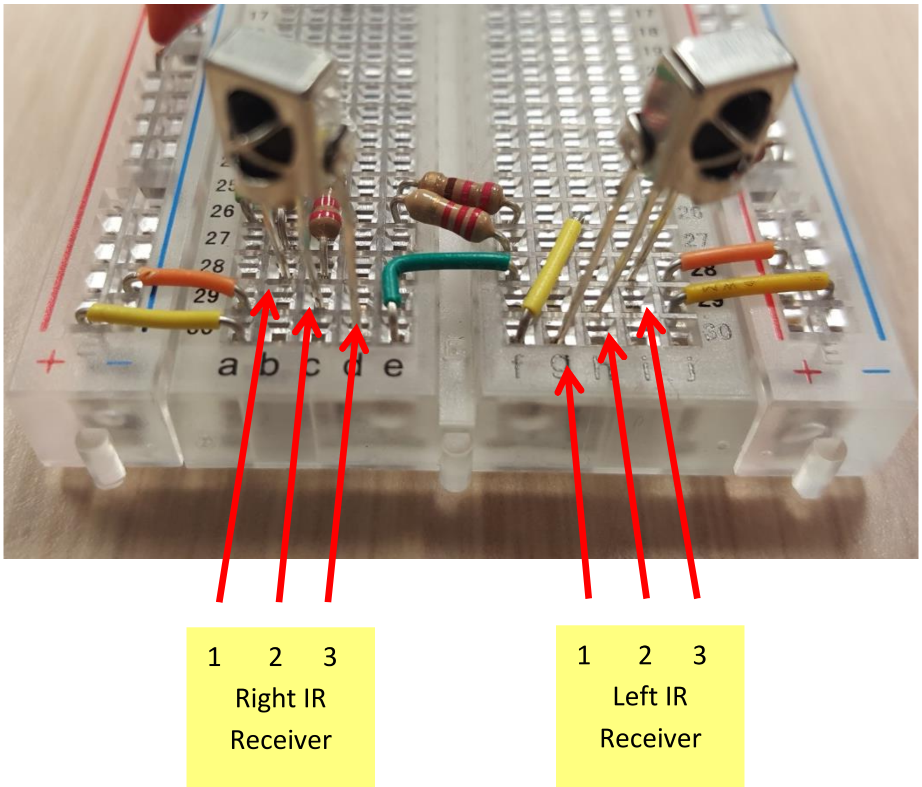
Long Lead is the Anode.

Short Lead is the Cathode.

Long Lead is the Anode.

14. Left IR1 Receiver pin 1 goes to g30, pin 2 goes to h29, and pin 3 goes to i28.

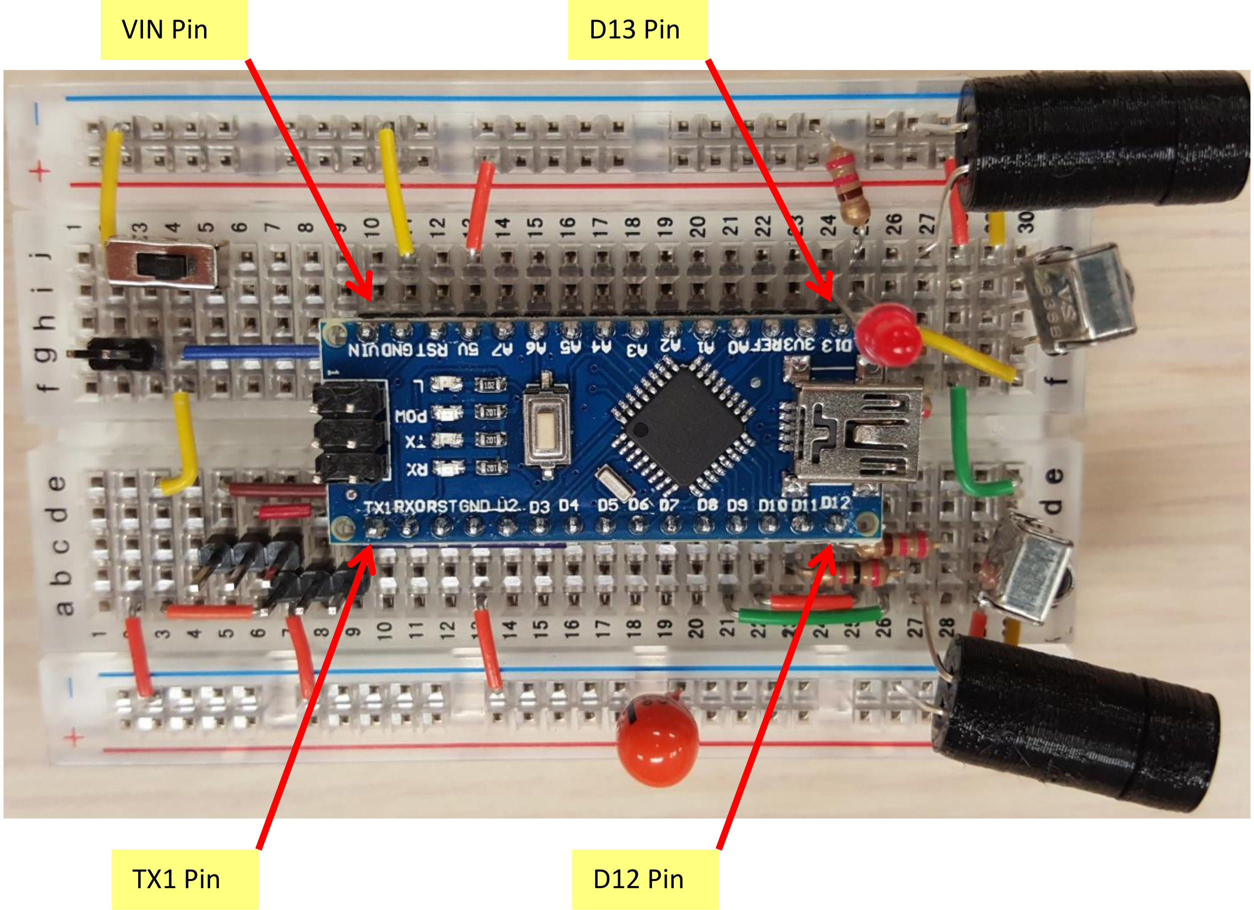
15. Right IR2 Receiver pin 1 goes to b28, pin 2 goes to c29, and pin 3 goes to d30.

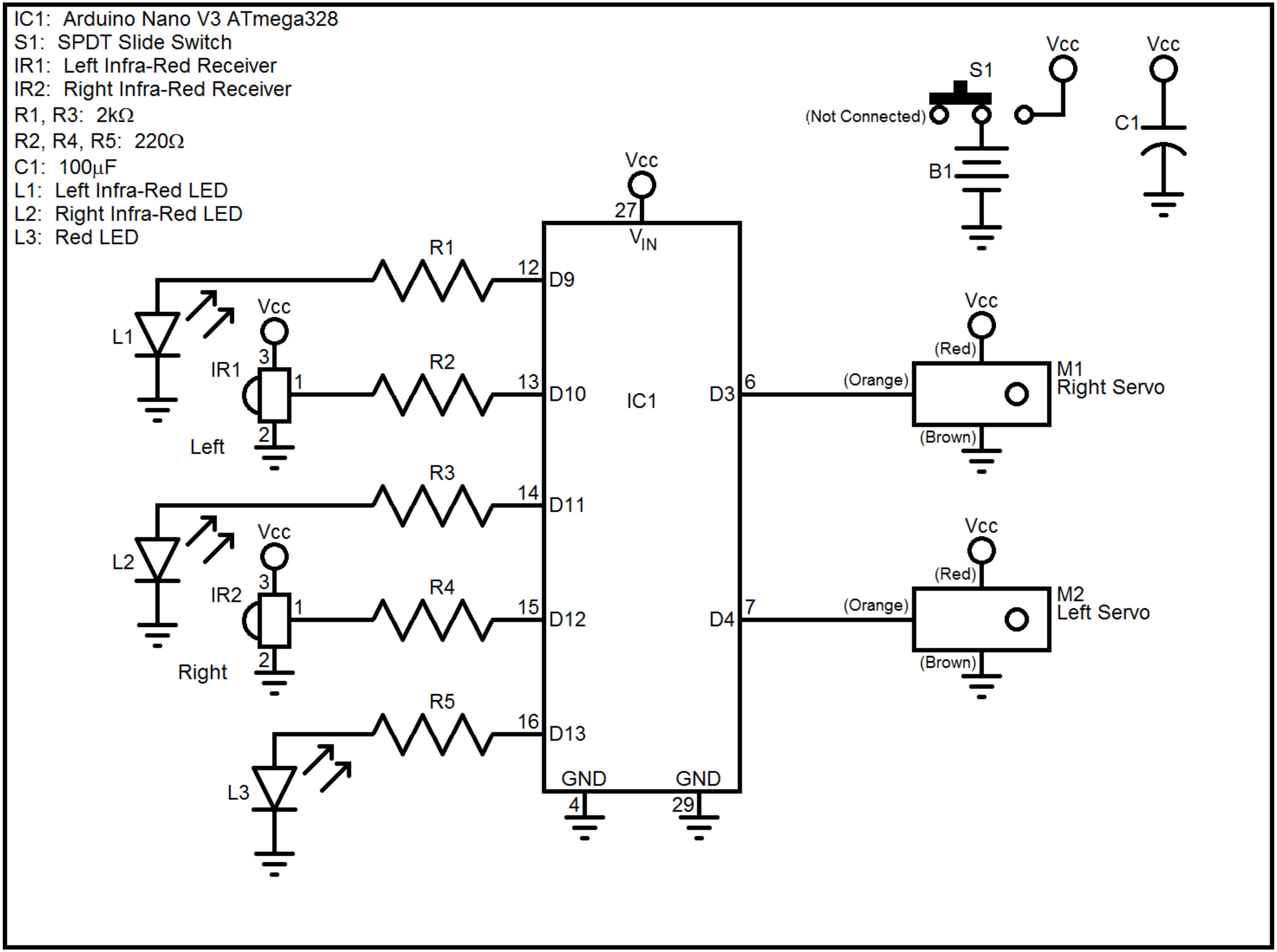


1 2 3
Right IR
Receiver

1 2 3
Left IR
Receiver

16. Arduino Nano V3 ATmega 328’s D13 pin goes in h24, VIN pin goes in h10, TX1 pin goes in d10, and D12 pin goes in d24.



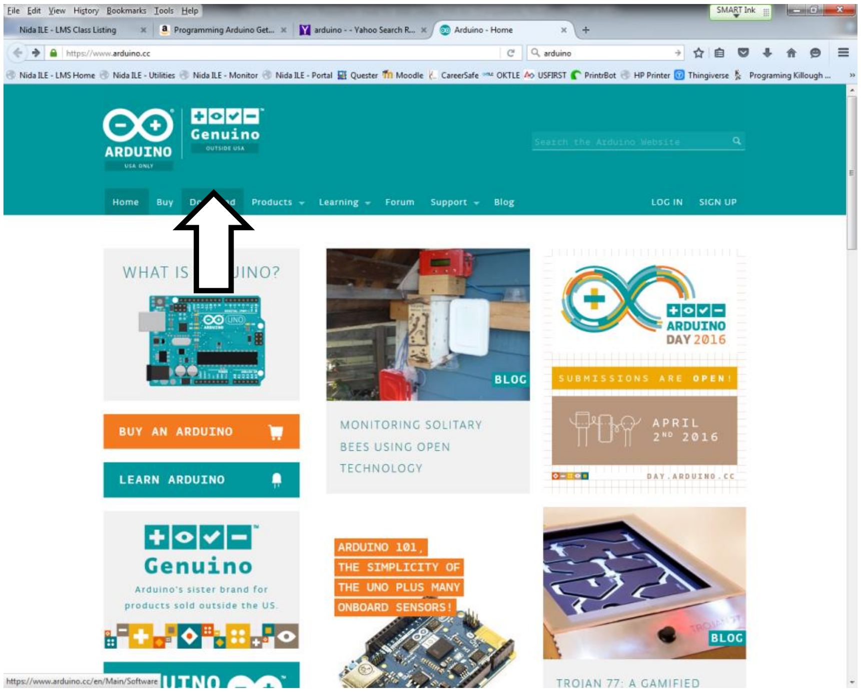


IC1: Arduino Nano V3 ATmega328
S1: SPDT Slide Switch
IR1: Left Infra-Red Receiver
IR2: Right Infra-Red Receiver
R1, R3: 2kΩ
R2, R4, R5: 220Ω
C1: 100µF
L1: Left Infra-Red LED
L2: Right Infra-Red LED
L3: Red LED

Part 2: Installing Software.

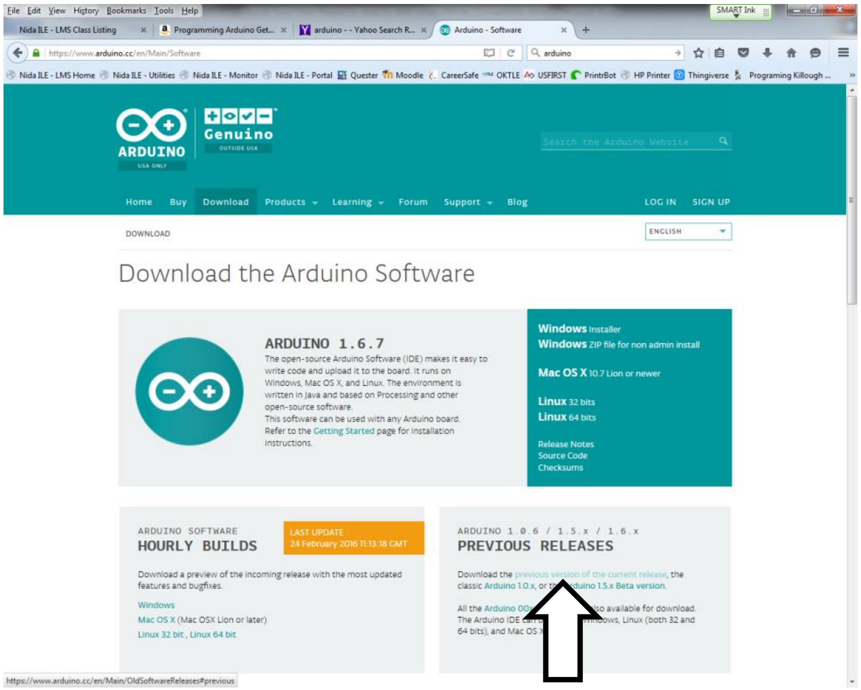
Go to www.arduino.cc

Click on Download.



Do NOT install the latest Arduino version.

Click on [previous version of the current release](https://www.arduino.cc/en/Main/OldSoftwareReleases#previous).


Arduino - Software
https://www.arduino.cc/en/Main/Software
ARDUINO
Genuino
Home
Buy
Download
Products
Learning
Forum
Support
Blog
LOG IN
SIGN UP
DOWNLOAD
ENGLISH
Windows Installer
Windows ZIP file for non admin install
Mac OS X 10.7 Lion or newer
Linux 32 bits
Linux 64 bits
Release Notes
Source Code
Checksums
LAST UPDATE 24 February 2016 11:13:18 GMT
Windows
Mac OS X (Mac OSX Lion or later)
Linux 32 bit , Linux 64 bit
https://www.arduino.cc/en/Main/OldSoftwareReleases#previous

Download the Arduino Software

ARDUINO 1.6.7

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for installation instructions.

ARDUINO SOFTWARE
HOURLY BUILDS

Download a preview of the incoming release with the most updated features and bugfixes.

ARDUINO 1.0.6 / 1.5.x / 1.6.x
PREVIOUS RELEASES

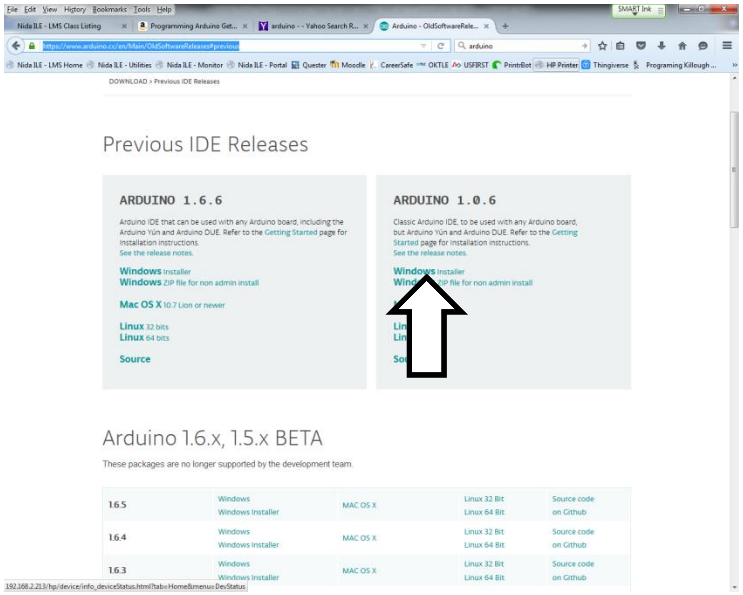
Download the previous version of the current release, the classic Arduino 1.0.x, or th rduino 1.5.x Beta version.

All the Arduino 00x lso available for download. The Arduino IDE ndows, Linux (both 32 and 64 bits), and Mac OS X

Click on the Windows installer for Arduino 1.0.6

This will start downloading the Arduino IDE installer.

For some reason, version 1.0.6 works with the Arduino Nano V3 and the later versions do not.


File Edit View History Bookmarks Tools Help
Nida ILE - LMS Class Listing
Programming Arduino Get...
arduino - - Yahoo Search R...
Arduino - OldSoftwareRele...
https://www.arduino.cc/en/Main/OldSoftwareReleases#previous
arduino
Nida ILE - LMS Home
Nida ILE - Utilities
Nida ILE - Monitor
Nida ILE - Portal
Quester
Moodle
CareerSafe
OKTLE
USFIRST
PrintrBot
HP Printer
Thingiverse
Programing Killough ...
SMART Ink

DOWNLOAD > Previous IDE Releases

Previous IDE Releases

ARDUINO 1.6.6

Arduino IDE that can be used with any Arduino board, including the Arduino Yún and Arduino DUE. Refer to the Getting Started page for installation instructions.
See the release notes.

Windows Installer
Windows ZIP file for non admin install

Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits

Source

ARDUINO 1.0.6

Classic Arduino IDE, to be used with any Arduino board, but Arduino Yún and Arduino DUE. Refer to the Getting Started page for installation instructions.
See the release notes.

Windows Installer
Wind ZIP file for non admin install

Lin
Lin

So

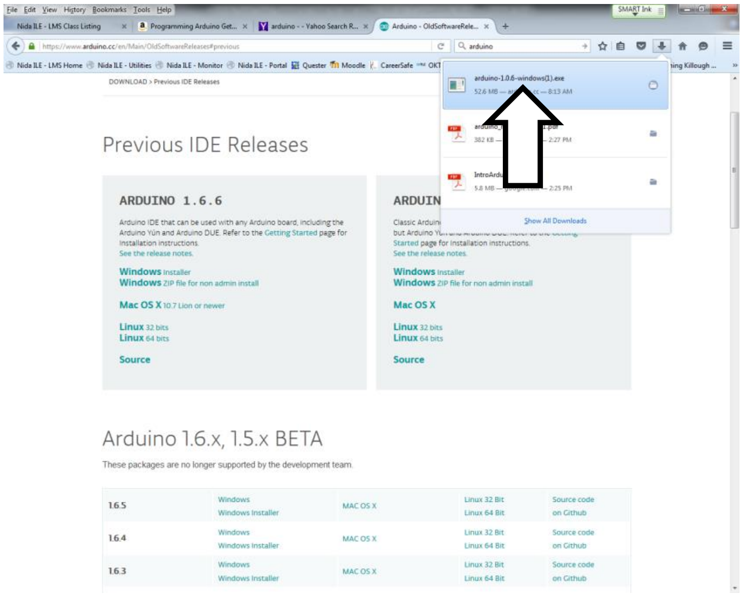
Arduino 1.6.x, 1.5.x BETA

These packages are no longer supported by the development team.

1.6.5	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.4	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.3	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github

192.168.2.213/hp/device/info_deviceStatus.html?tab=Home&menu=DevStatus

Install Software by clicking on the file you just downloaded and follow any instructions the installer gives you.


File Edit View History Bookmarks Tools Help
Nida ILE - LMS Class Listing
Programming Arduino Get...
arduino - - Yahoo Search R...
Arduino - OldSoftwareRele...
https://www.arduino.cc/en/Main/OldSoftwareReleases#previous
arduino
SMART Ink
Show All Downloads

arduino-1.0.6-windows(1).exe
52.6 MB — ar cc — 8:13 AM

arduino_ .pdf
382 KB — — 2:27 PM

IntroArdu
5.8 MB — — 2:25 PM

DOWNLOAD > Previous IDE Releases

Previous IDE Releases

ARDUINO 1.6.6

Arduino IDE that can be used with any Arduino board, including the Arduino Yún and Arduino DUE. Refer to the Getting Started page for installation instructions.
See the release notes.

Windows Installer
Windows ZIP file for non admin install

Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits

Source

ARDUIN

Classic Arduin
but Arduino Y
Started page for installation instructions.
See the release notes.

Windows Installer
Windows ZIP file for non admin install

Mac OS X

Linux 32 bits
Linux 64 bits

Source

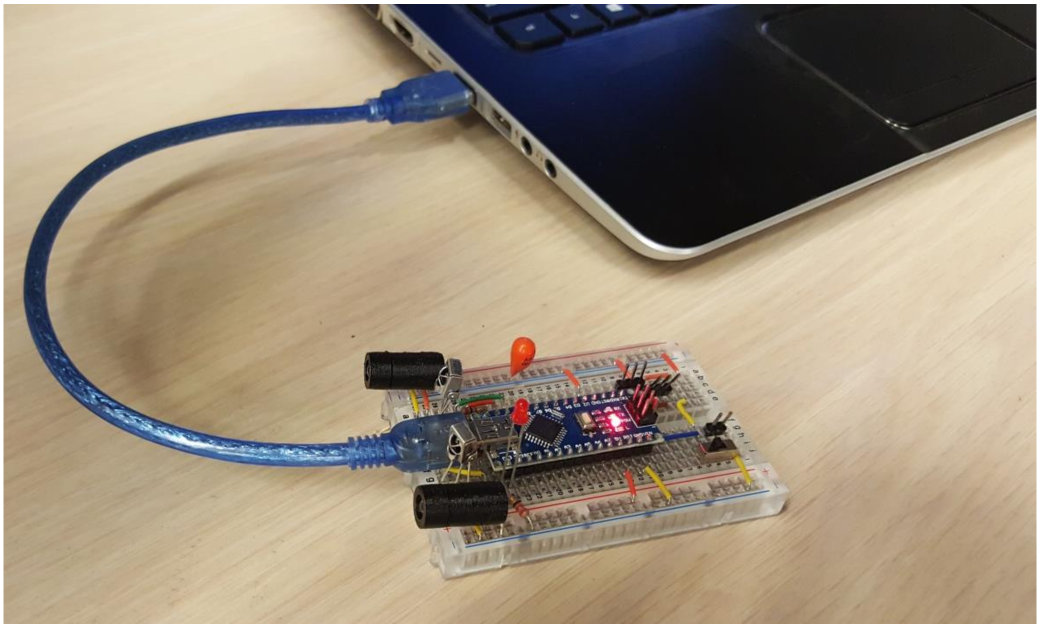
Arduino 1.6.x, 1.5.x BETA

These packages are no longer supported by the development team.

1.6.5	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.4	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.3	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github

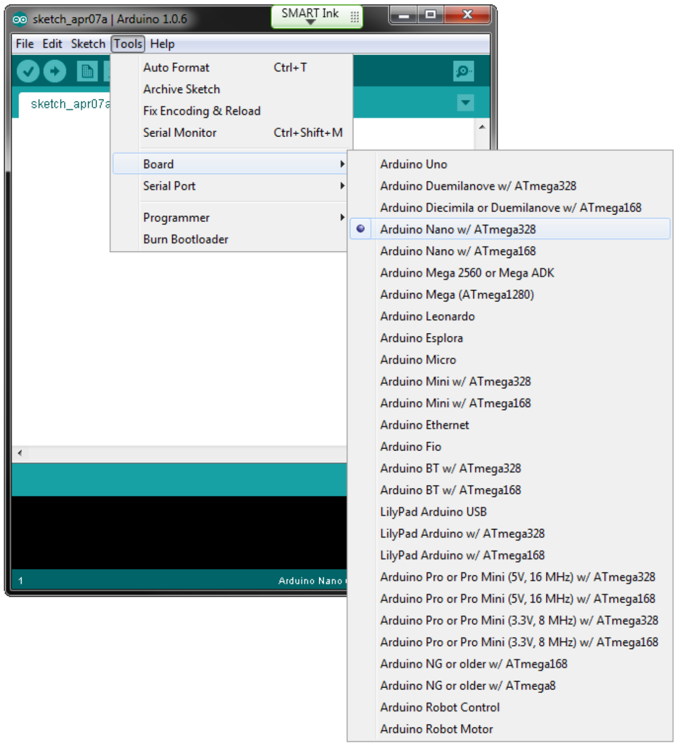
Make note about installing the USB driver

NOTE: After the Arduino IDE is installed, there is a couple of settings that need to be changed.



First, connect the mini USB cable between the mini USB port on the Arduino Nano and the USB port on the PC.

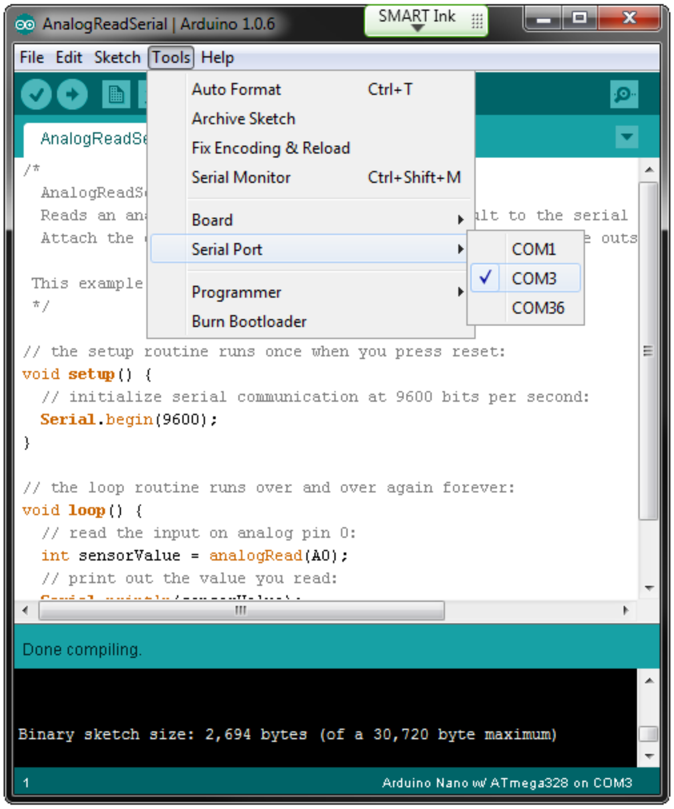
Click on Tools →Board →Arduino Nano w/ ATmega328.


sketch_apr07a | Arduino 1.0.6
SMART Ink
File
Edit
Sketch
Tools
Help
sketch_apr07a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor
1
Arduino Nano

Also, to send code to the Arduino and use the Serial Monitor, you must have the right serial Port Connected.

Click on Tools →Serial Port →and select the COM port that represents the Arduino Nano

NOTE: This might be tricky. If you select one and it will not upload any code, select a different COM port until the code will upload.



```
/*
  AnalogReadS
  Reads an an                          lt to the serial
  Attach the                           outs

 This example
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
```

```
Done compiling.

Binary sketch size: 2,694 bytes (of a 30,720 byte maximum)
```

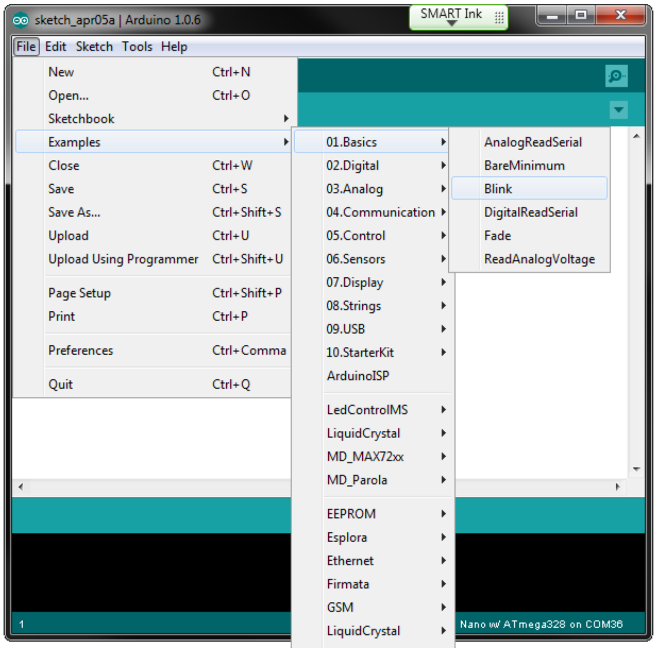
Activity 1: Load a Sketch

Before we attach the circuit board to the ServoBot's Chassis, we are going to test it using the Blink sketch. This sketch is already written for us and located in the Arduino library.

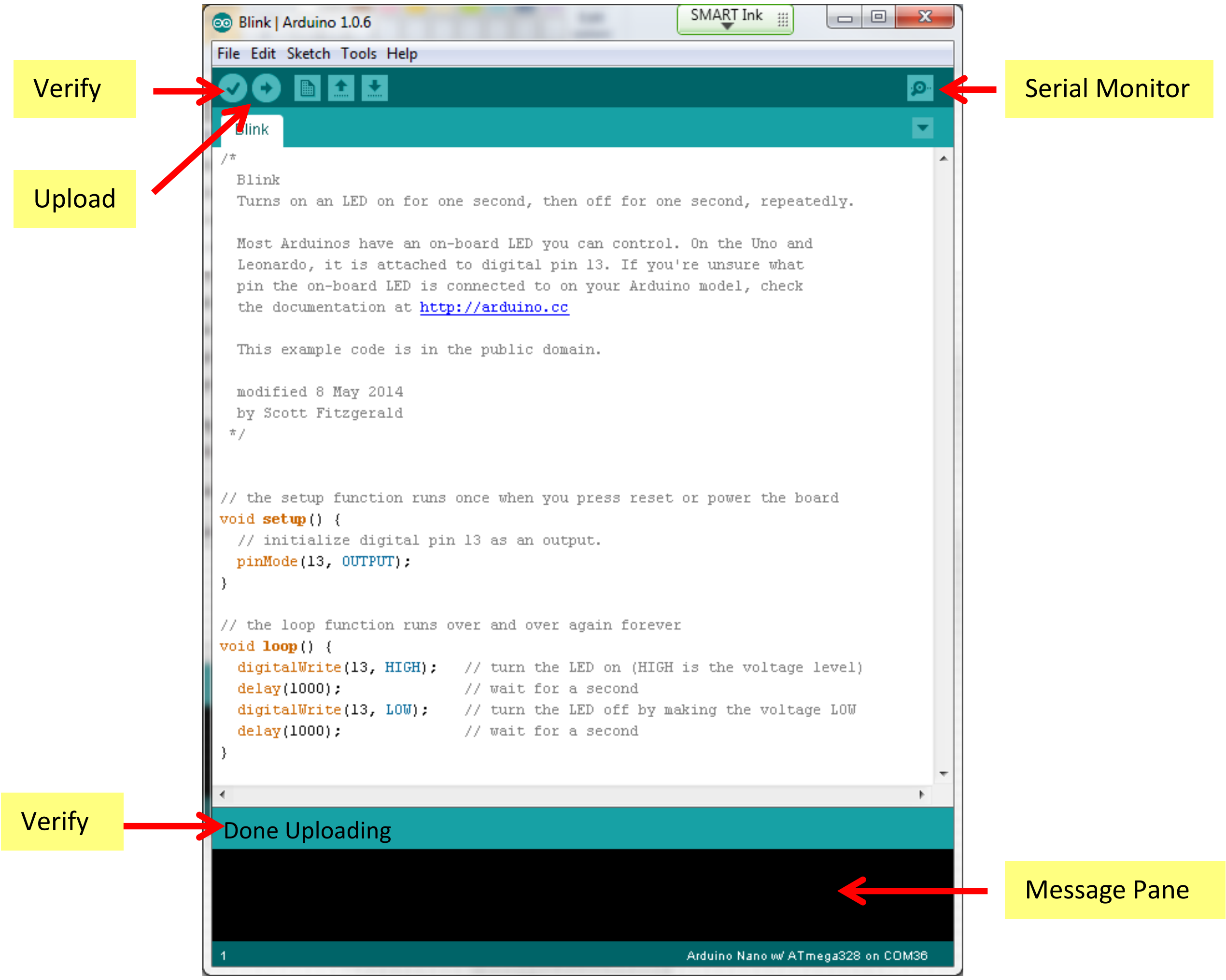
To access it:

Load the Arduino Development Environment (IDE)

Click on File →Examples →01.Basic →Blink.



This will load the Blink sketch. (It should look like the screenshot below.)



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://arduino.cc

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald
 */


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Done Uploading

1   Arduino Nano w/ ATmega328 on COM36

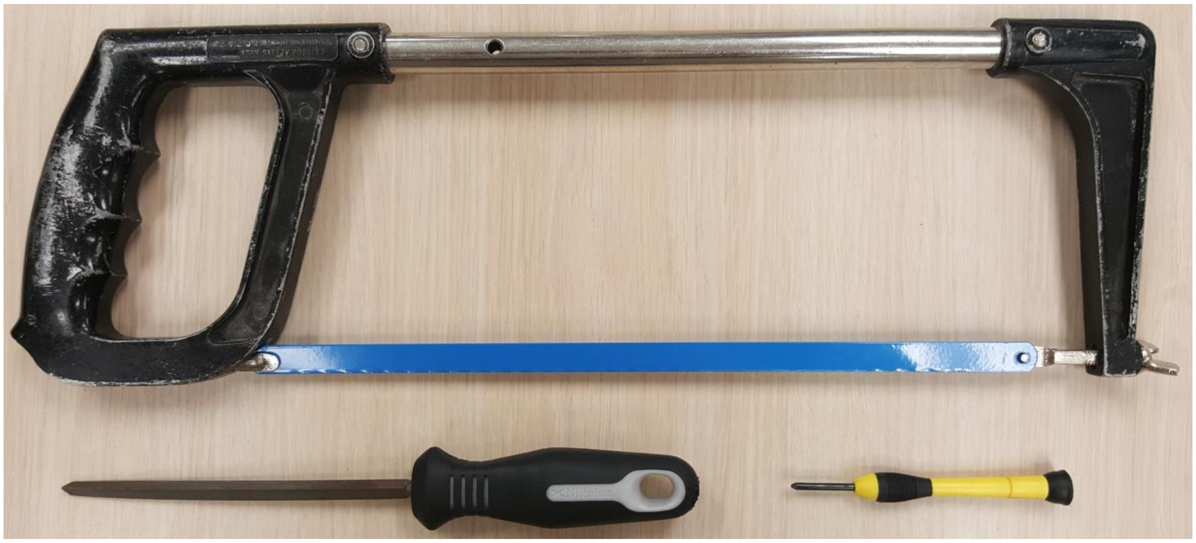
- ✓ Click the Verify button to make sure your code doesn’t have any typing errors.
- ✓ Look for the “Binary sketch size” text in the message pane.
    - If it’s there, your code compiled and is ready to upload to the Arduino.
    - If there’s a list of errors instead, it’s trying to tell you it can’t compile your code. So, find the typing mistake and fix it!
- ✓ Click the Upload button. The status line under your code will display “Compiling sketch…,” “Uploading…,” and then “Done uploading.”
- ✓ Verify that the red LED is blinking on and off at 1 second intervals.
- ✓ Modify the `delay(1000);` instructions to 3000. This will make the red LED blink slower.
- ✓ Verify that the red LED is blinking on and off at 3 second intervals.

Activity 2: Write a "Hello!" Sketch

```
/*
Robotics with the ServoBot – Hello
Have the Arduino say, “Hello.”
*/

void setup()

{
   Serial.begin(9600);
   Serial.print("Hello!");
}

void loop()
{
   //Add code that repeats automatically here.
}
```

Part 3: Robot Construction.

Tools Needed:



Hack Saw

Jeweler’s screwdriver set w/ #1 Philips Screwdriver

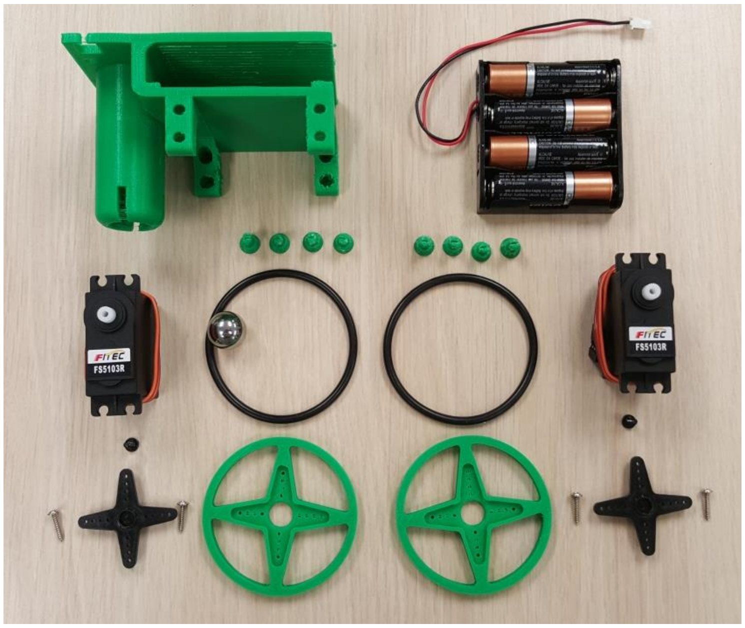
Smooth cut metal file

You will also need Safety Glasses.

NOTE: You will need to wear safety glasses while using the hack saw in case bits of metal fly at your eyes when cutting. Make sure the glasses cover the sides of your eyes. If you wear glasses, get safety glasses that will fit over your regular glasses comfortably. Instead of safety glasses, you may install side shields on your regular glasses and use them if they fully cover your eyes.

NOTE: Make sure to bind any loose clothing or hair and remove any jewelry.

Parts Needed for Constructing the Body of the ServoBot:



3D Printed Robot Chassis

3D Printed Wheels (X2)

3D Printed Mechanical Fasteners (X8)

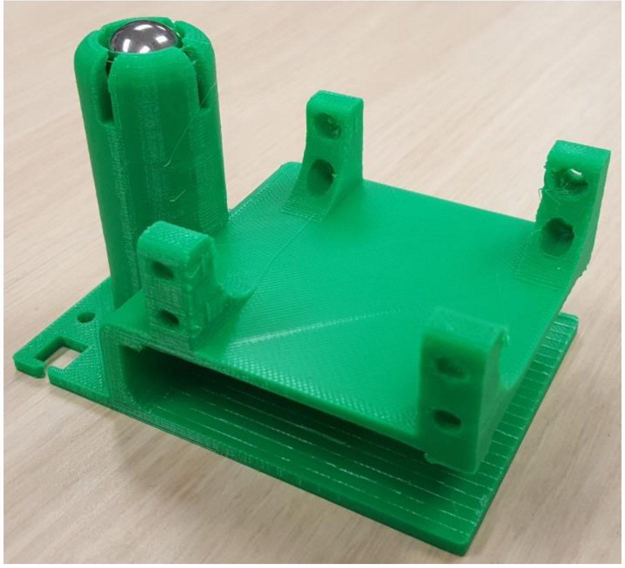
Quad AA Battery Holder

AA Batteries (X4)

FS5103R Continuous Rotation Servos w 4 point horn and mounting screws(X2)

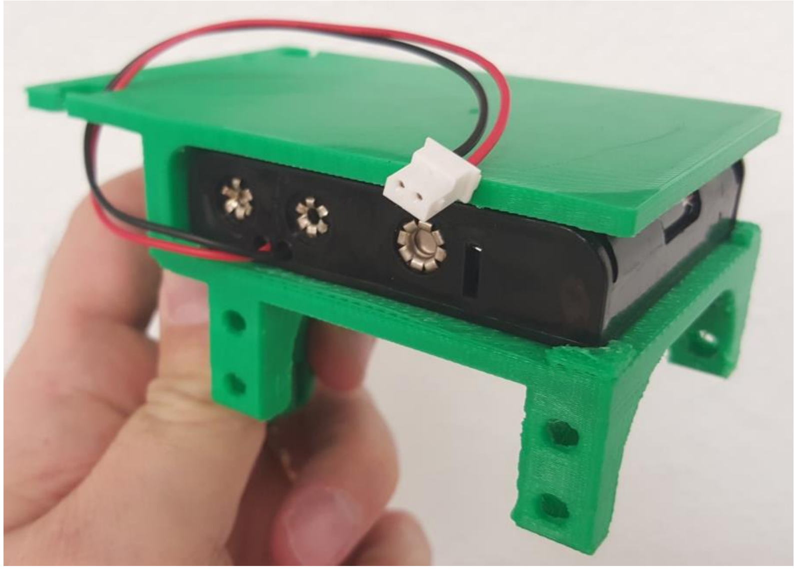
Black O-ring.

5/8” Ball Bearing



1. Take the 5/8” Ball Bearing and place it in the round shaft on the robot chassis.

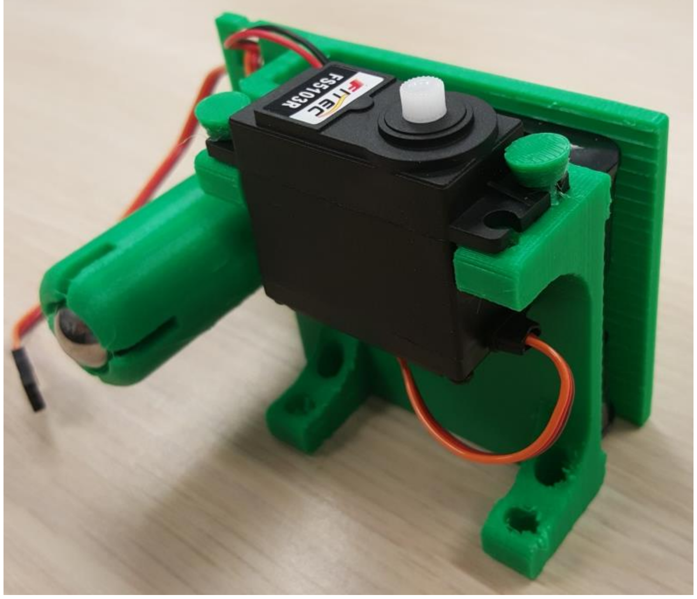
2. Take 4 AA batteries and place them in the battery holder.



3. Insert the battery holder in the robot chassis as pictured above.

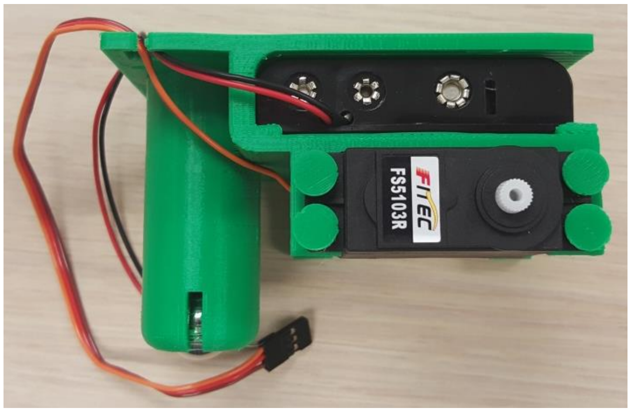
4. Insert the wire from the battery holder into square hole located on the back of the chassis.

5. ***NOTE: Make sure the wire coming from the battery holder is on the right side of the chassis.***



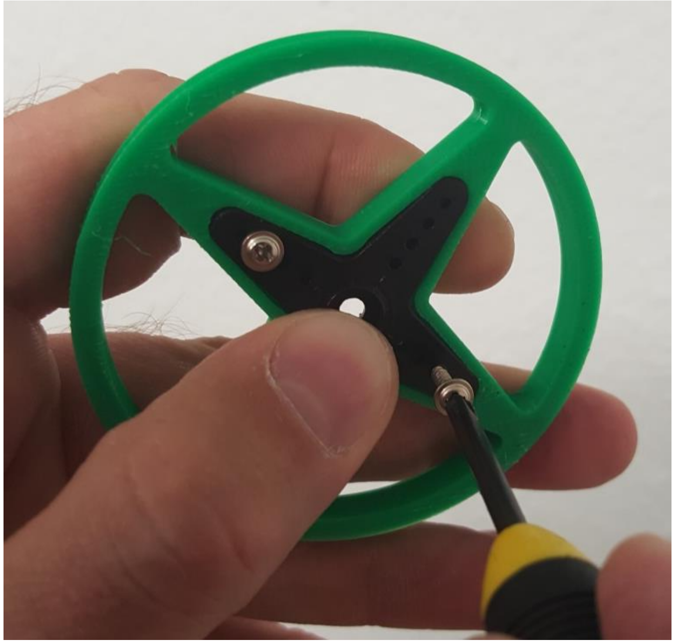
6. Install the servo into the bottom slot.

7. ***NOTE: Make sure the servo's wire is facing towards the front of the robot. This will make the servo's calibration screw easily accessible.***



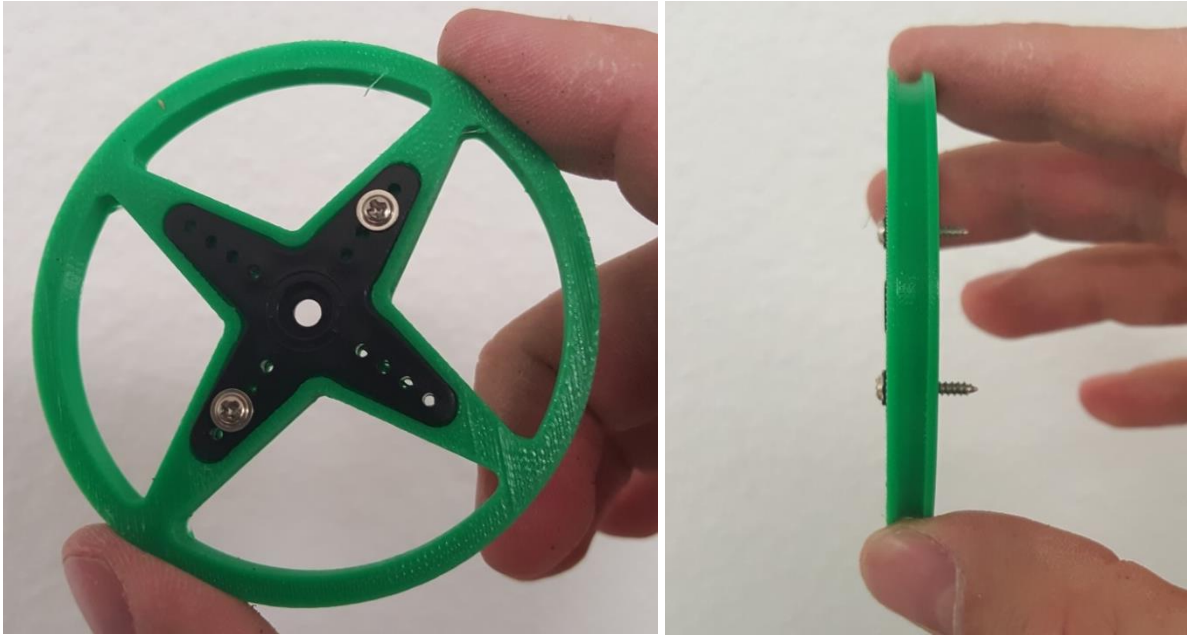
8. Secure the servo with 4 3D printed pins.

9. ***NOTE: Make sure the 3D printed pins are filed before inserting them. DO NOT FOCE them into the holes. It might split the plastic of the chassis. The 3D printed pins might need some filing to fit properly. .***



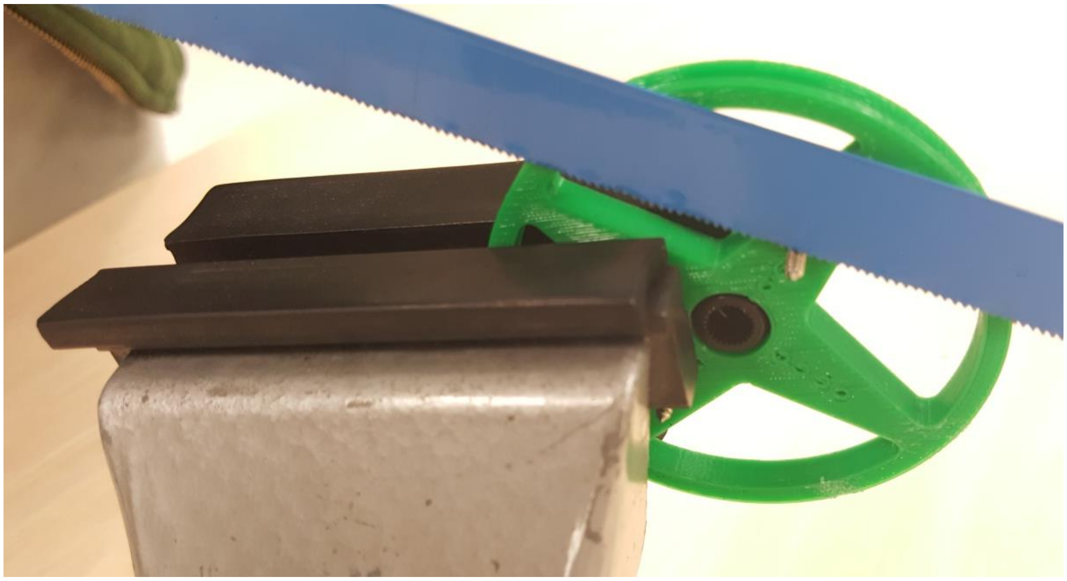
10. Take the four point servo horn that came with the servo and place it in the recessed slot on the wheel.

11. Using the #1 Philips screwdriver, insert the screws came with the servo in the second to last hole on opposite ends of the servo horn.



12. When finished, the servo horn should be flush with the will on both sides when properly screwed in.

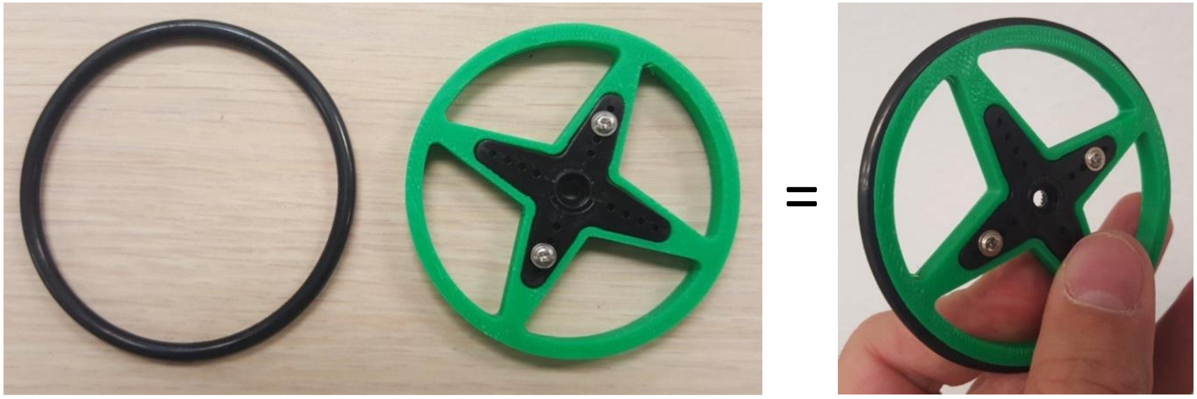
13. ***NOTE: It may be necessary to pre screw each hole in the wheel and the servo horn before attaching them together to ensure they are flush. ALSO NOTE the screws sticking out the back.***



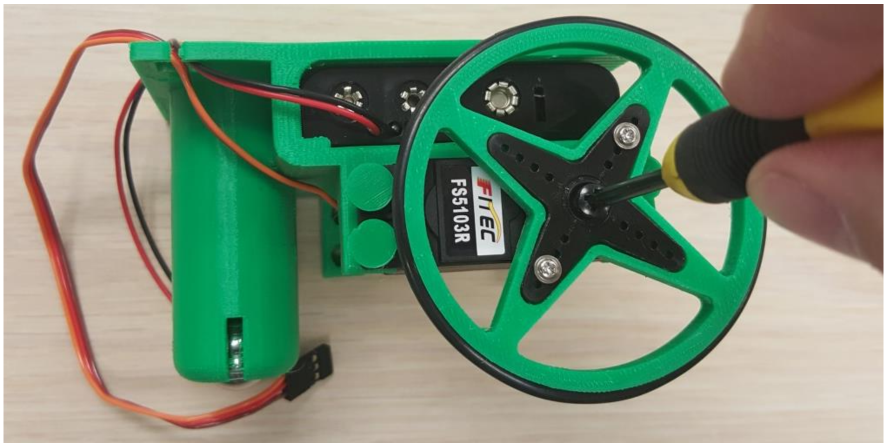
14. Before installing the wheel on the servo, cut the screw so that it is flush with the backside of the wheel.

15. NOTE: USE Safety Glasses while cutting. Also, it might be necessary to use a vise to hold the wheel while cutting the screw. DO NOT have your fingers close to the hack saw blade. ***

16. After cutting, file the ends of the screws smooth to reduce the chances they might puncture skin.

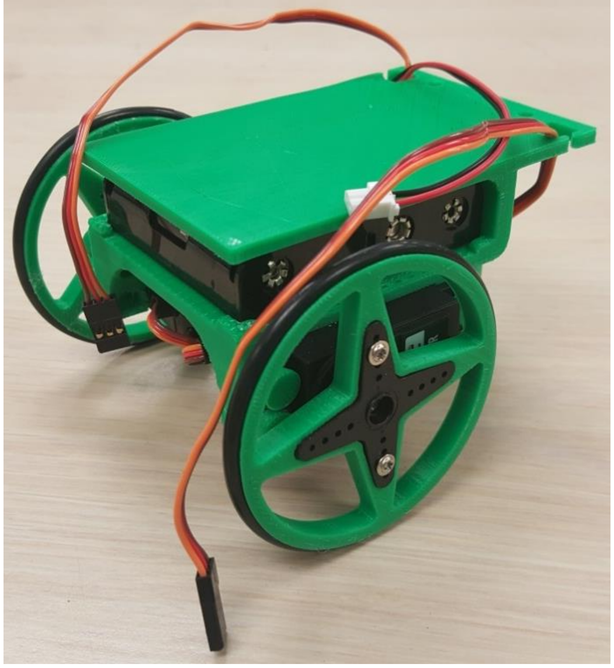


17. Take the black O-ring and place it around the wheel.

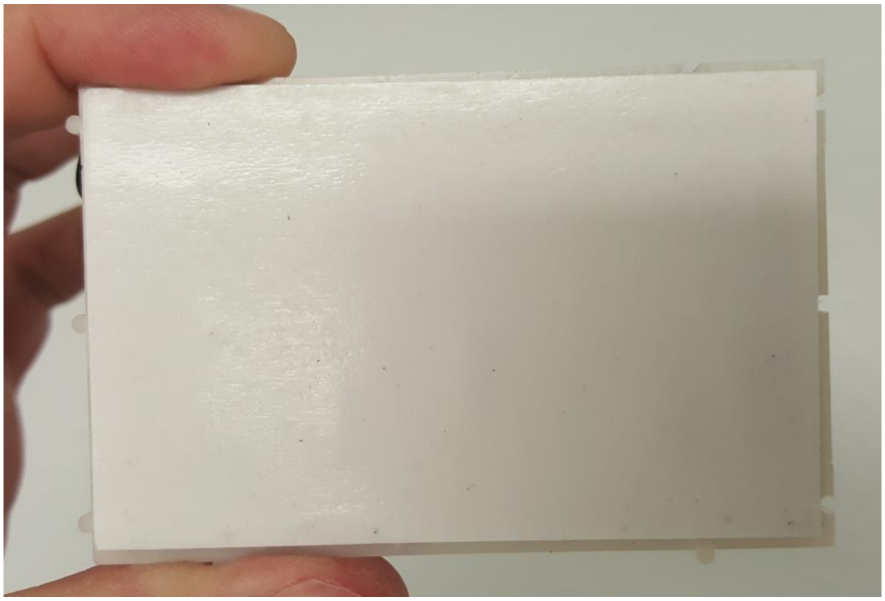


19. Then take the black screw that came packaged with the servo and screw the wheel/ servo horn assembly to the serve.

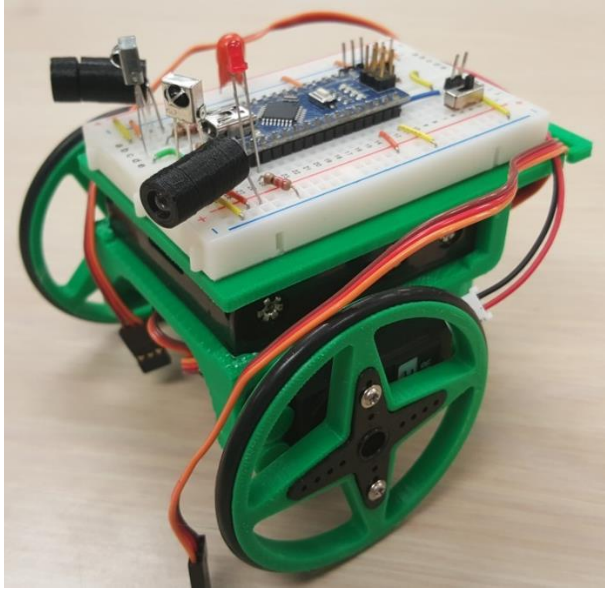
20. Repeat these steps for the other servo.

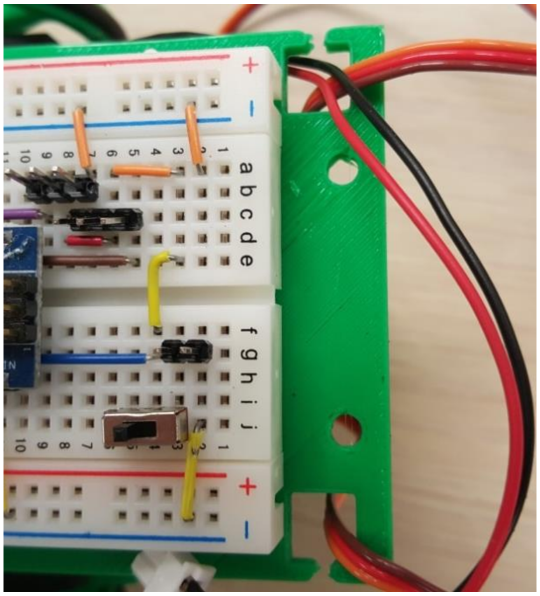


21. After both servos are installed, take the servo wires and thread them through the rear square holes for cable management.



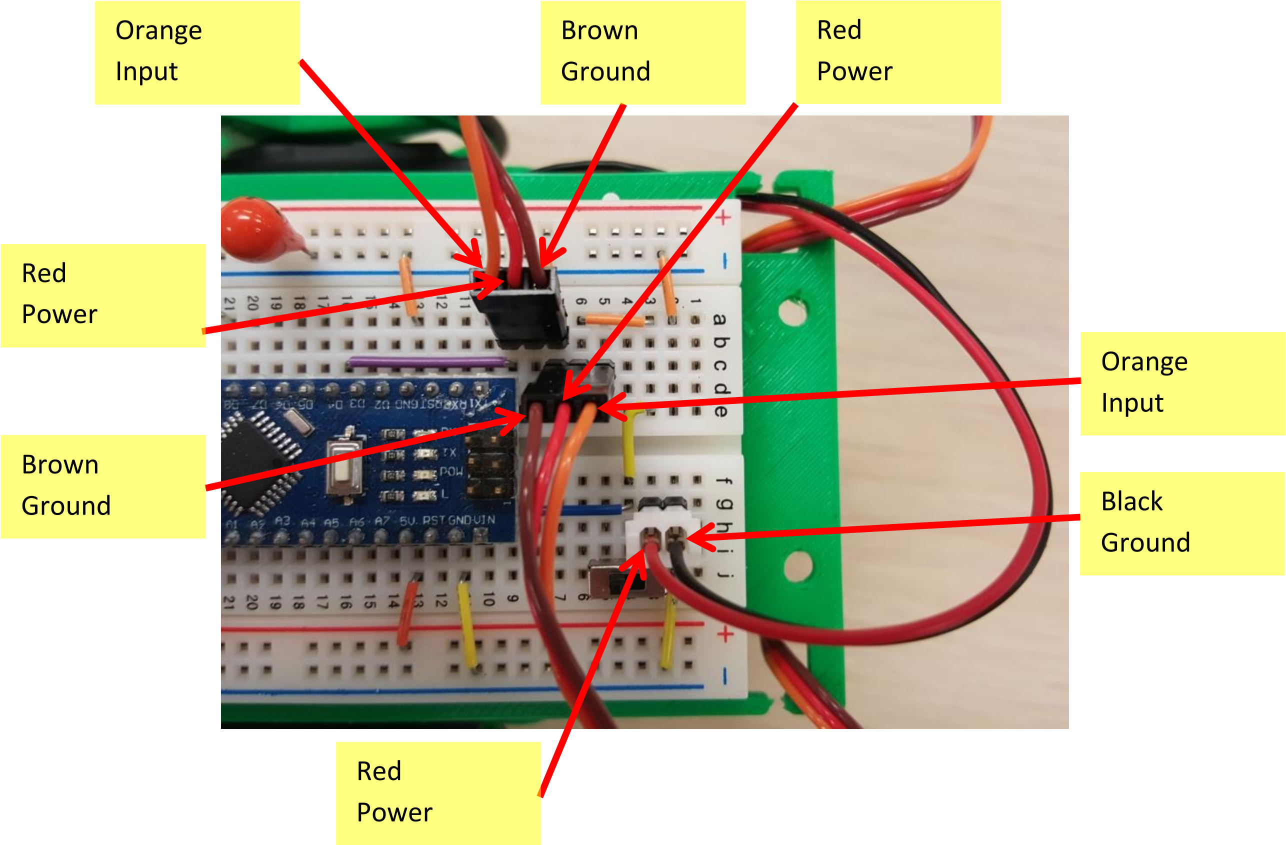
22. Take the protoboard and remove the paper from the double sided tape.



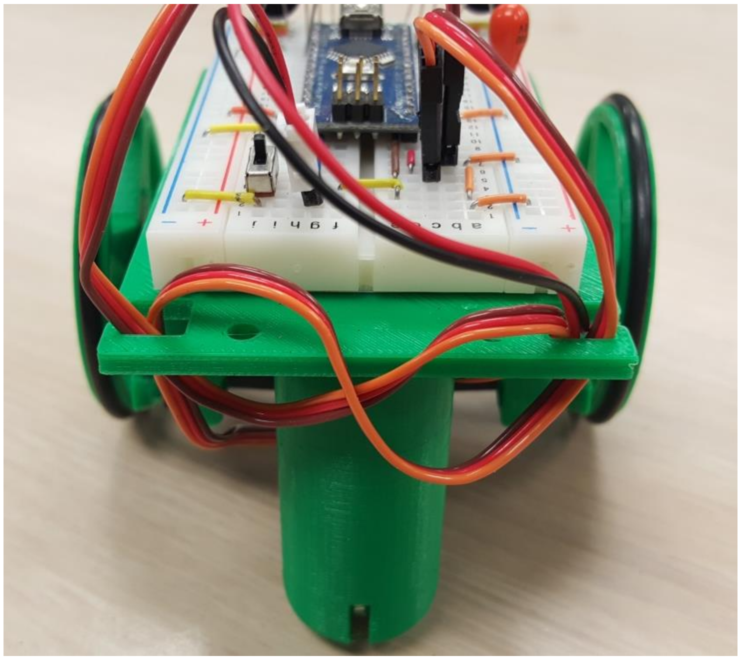


23. CAREFULLY stick the breadboard to the top of the robot chassis.

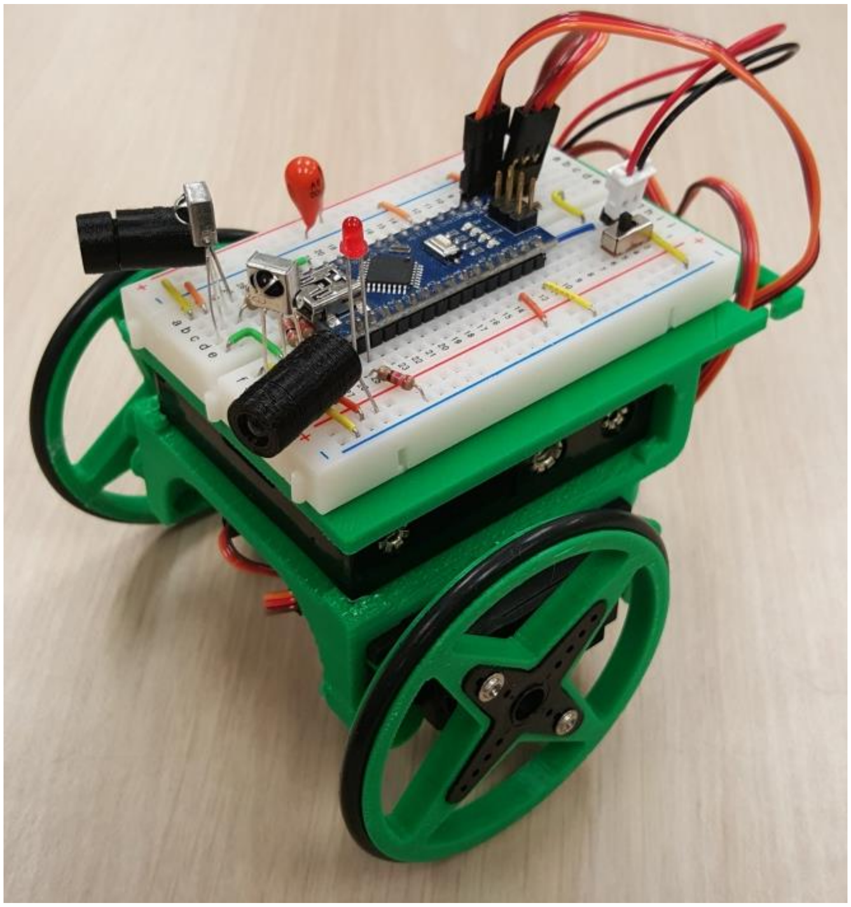
24. ***NOTE: Make sure the rear square holes on the back of the chassis are not covered with the protoboard. Refer to the pictures above for clarification.***



26. Attach the servo and battery wires as shown above.



27. Finally, crisscross the left and right servo wires in the rear square holes to take up the slack in the servo wires. Refer to the picture above for clarification.



The Finished ServoBot.

Part 4. Programming.

Activity 3: Centering the Servos

Example Sketch: ServoCal

The code below allows the user to center both servos at the same time instead of one.

```
/*
Robotics with the ServoBot – ServoCal
Transmit the center or stay still signals on pin 3 and 4 for center
adjustment.
*/

#include <Servo.h>                       // Include servo library

Servo servoRight;                        // Declare right servo
Servo servoLeft;                         // Declare left servo

void setup()                             // Built-in initialization block
{
   servoRight.attach(3);                 // Attach right signal to pin 3
   servoLeft.attach(4);                  // Attach left signal to pin 4

   servoRight.writeMicroseconds(1500); // 1.5 ms stay still signal
   servoLeft.writeMicroseconds(1500);  // 1.5 ms stay still signal
}

void loop()                              // Main loop auto-repeats
{                                        // Empty, nothing needs repeating
}
```

Activity 4: Controlling Servo Speed, Direction & Run Time

Example Sketch: ServoTest

The code below allows the user to test both servos at the same time instead of one.

```
/*
Robotics with the ServoBot – ServoTest
Transmit the center or stay still signals on pin 3 and 4 for center
adjustment.
*/

#include <Servo.h>                        // Include servo library

Servo servoRight;                         // Declare right servo
Servo servoLeft;                          // Declare left servo

void setup()                              // Built-in initialization block
{
   servoRight.attach(3);                  // Attach right signal to pin 3
   servoLeft.attach(4);                   // Attach left signal to pin 4

   servoRight.writeMicroseconds(1300);    // Right Servo spins clockwise
   servoLeft.writeMicroseconds(1300);     // Left Servo spins clockwise
   delay(3000);                           // ..for 3 seconds

   servoRight.writeMicroseconds(1500);    // Right Servo stops
   servoLeft.writeMicroseconds(1500);     // Left Servo stops
   delay(3000);                           // ..for 3 seconds

   servoRight.writeMicroseconds(1700);    // Right Servo spins counterclockwise
   servoLeft.writeMicroseconds(1700);     // Left Servo spins counterclockwise
   delay(3000);                           // ..for 3 seconds
   servoRight.detach();                   // Stop sending servo signal
   servoLeft.detach();                    // Stop sending servo signal
}

void loop()                               // Main loop auto-repeats
{                                         // Empty, nothing needs repeating
}
```

Activity 5: Going Forward and Reverse

Example Sketch: Servo_Forward_Reverse

The code below makes the ServoBot go forward 3 seconds, stop 3 seconds, reverse 3 seconds, then stop permanently.

```
/*
Robotics with the ServoBot - Servo_Forward_Reverse
Transmit the center or stay still signals on pin 3 and 4 for center
adjustment.
*/

#include <Servo.h>                        // Include servo library

Servo servoRight;                         // Declare right servo
Servo servoLeft;                          // Declare left servo

void setup()                              // Built-in initialization block
{
   servoRight.attach(3);                  // Attach right signal to pin 3
   servoLeft.attach(4);                   // Attach left signal to pin 4

//Forward
   servoRight.writeMicroseconds(1300);    // Right Servo spins clockwise
   servoLeft.writeMicroseconds(1700);     // Left Servo spins clockwise
   delay(3000);                           // ..for 3 seconds

//Stop
   servoRight.writeMicroseconds(1500);    // Right Servo stops
   servoLeft.writeMicroseconds(1500);     // Left Servo stops
   delay(3000);                           // ..for 3 seconds

//Reverse
   servoRight.writeMicroseconds(1700);    // Right Servo spins counterclockwise
   servoLeft.writeMicroseconds(1300);     // Left Servo spins counterclockwise
   delay(3000);                           // ..for 3 seconds

   servoRight.detach();                   // Stop sending servo signal
   servoLeft.detach();                    // Stop sending servo signal
}
void loop()                               // Main loop auto-repeats
{                                         // Empty, nothing needs repeating
}
```

Want to change the distance traveled? Just change the time in delay(3000). For example, delay(1500) will make the ServoBot go for only half the time, which in turn will make it travel only half as far. Likewise, delay(6000) will make it go for twice the time, and therefore twice the distance.

Activity 6: Turning Left, Right, and Pivoting

Example Sketch: ForwardRightLeftBackward

The code below makes the ServoBot go forward 3 seconds, stop 3 seconds, turn right in place 3 seconds, stop 3 seconds, turn left in place 3 seconds, stop 3 seconds, reverse 3 seconds, stop 3 seconds, pivot right forward 3 seconds, stop 3 seconds, pivot left forward 3 seconds, stop 3 seconds, pivot right reverse 3 seconds, stop 3 seconds, pivot left reverse 3 seconds, and then stop permanently.

```
/*
Robotics with the ServoBot – ForwardRightLeftBackward
Transmit the center or stay still signals on pin 3 and 4 for center
adjustment.
*/

#include <Servo.h>                        // Include servo library

Servo servoRight;                         // Declare right servo
Servo servoLeft;                          // Declare left servo

void setup()                              // Built-in initialization block
{
   servoRight.attach(3);                  // Attach right signal to pin 3
   servoLeft.attach(4);                   // Attach left signal to pin 4

//Forward
   servoRight.writeMicroseconds(1300);   // Right Servo spins clockwise
   servoLeft.writeMicroseconds(1700);    // Left Servo spins clockwise
   delay(3000);                           // ..for 3 seconds

//Stop
   servoRight.writeMicroseconds(1500);   // Right Servo stops
   servoLeft.writeMicroseconds(1500);    // Left Servo stops
   delay(3000);                           // ..for 3 seconds

//Turn Right in place
   servoRight.writeMicroseconds(1300);   // Right Servo spins clockwise
   servoLeft.writeMicroseconds(1300);    // Left Servo spins counterclockwise
   delay(3000);                           // ..for 3 seconds

//Stop
   servoRight.writeMicroseconds(1500);   // Right Servo stops
   servoLeft.writeMicroseconds(1500);    // Left Servo stops
   delay(3000);                           // ..for 3 seconds

//Turn Left in place
   servoRight.writeMicroseconds(1700);   // Right Servo spins counterclockwise
   servoLeft.writeMicroseconds(1700);    // Left Servo spins clockwise
   delay(3000);                           // ..for 3 seconds

//Stop
   servoRight.writeMicroseconds(1500);   // Right Servo stops
   servoLeft.writeMicroseconds(1500);    // Left Servo stops
   delay(3000);                           // ..for 3 seconds

//Reverse
```

```
   servoRight.writeMicroseconds(1700);   // Right Servo spins counterclockwise
   servoLeft.writeMicroseconds(1300);    // Left Servo spins counterclockwise
   delay(3000);                          // ..for 3 seconds

// Pivot forward-left
   servoRight.writeMicroseconds(1300);   // Right Servo spins clockwise
   servoLeft.writeMicroseconds(1500);    // Left Servo stops

// Pivot forward-right
   servoRight.writeMicroseconds(1500);   // Right Servo stops
   servoLeft.writeMicroseconds(1700);    // Left Servo spins clockwise

// Pivot backward-left
   servoRight.writeMicroseconds(1700);   // Right Servo spins counterclockwise
   servoLeft.writeMicroseconds(1500);    // Left Servo stops

// Pivot backward-right
   servoRight.writeMicroseconds(1500);   // Right Servo stops
   servoLeft.writeMicroseconds(1300);    // Left Servo spins counterclockwise

   servoRight.detach();                  // Stop sending servo signal
   servoLeft.detach();                   // Stop sending servo signal
}

void loop()                              // Main loop auto-repeats
{                                        // Empty, nothing needs repeating
}
```

Activity 7: Put Maneuvers into Functions

Example Sketch – MovementsWithSimpleFunctions

The code below makes the ServoBot go forward 2 seconds, stop 0.5 seconds, turn right 0.6 seconds, stop 0.5 seconds, turn left 0.6 seconds, stop 0.5 seconds, reverse 2 seconds, and then stop permanently.

```
/*
Robotics with the ServoBot - MovementsWithSimpleFunctions
Transmit the center or stay still signals on pin 3 and 4 for center
adjustment.
*/

#include <Servo.h>                       // Include servo library

Servo servoRight;                        // Declare right servo
Servo servoLeft;                         // Declare left servo

void setup()                             // Built-in initialization block
{
   servoRight.attach(3);                 // Attach right signal to pin 3
   servoLeft.attach(4);                  // Attach left signal to pin 4

   forward(2000);                        //go forward for 2 seconds
   stop(500);                            //stops for 0.5 seconds
   turnRight(600);                       //turn right for 0.6 seconds
   stop(500);                            //stops for 0.5 seconds
   turnLeft(600);                        //turn left for 0.6 seconds
   stop(500);                            //stops for 0.5 seconds
   reverse(2000);                        //go backward for 2 seconds

   disableServos();                      //stay still indefinitely

   servoRight.detach();                  // Stop sending servo signal
   servoLeft.detach();                   // Stop sending servo signal
}

void loop()                              // Main loop auto-repeats
{                                        // Empty, nothing needs repeating
}

void forward(int time)
{
   servoRight.writeMicroseconds(1300);   // Right Servo spins clockwise
   servoLeft.writeMicroseconds(1700);    // Left Servo spins clockwise
   delay(time);                          // ..Maneuver for time ms
}
void turnRight(int time)
{
   servoRight.writeMicroseconds(1300);   // Right Servo spins clockwise
   servoLeft.writeMicroseconds(1300);    // Left Servo spins counterclockwise
   delay(time);                          // ..Maneuver for time ms
}
void turnLeft(int time)
{
   servoRight.writeMicroseconds(1700);   // Right Servo spins counterclockwise
   servoLeft.writeMicroseconds(1700);    // Left Servo spins clockwise
   delay(time);                          // ..Maneuver for time ms
}
void reverse(int time)
{
   servoRight.writeMicroseconds(1700);   // Right Servo spins counterclockwise
   servoLeft.writeMicroseconds(1300);    // Left Servo spins counterclockwise
   delay(time);                          // ..Maneuver for time ms
}
void stop(int time)
{
   servoRight.writeMicroseconds(1500);   // Right Servo stops
   servoLeft.writeMicroseconds(1500);    // Left Servo stops
   delay(time);                          // ..Maneuver for time ms
}
void disableServos()
{
   servoRight.detach();                  // Stop sending servo signal
   servoLeft.detach();                   // Stop sending servo signal
}
```

You should recognize the pattern of movement your ServoBot makes. Change up the Function Calls to make the ServoBot move in different patterns such as a square or triangle.

Activity 8: Object Detection Test Code

Example Sketch: TestIR

The code below tests both sets of IR LED’s and Receivers instead of one at a time.

After uploading the code to the ServoBot, keep the USB cord connected between the computer and ServoBot. Then open the Serial Monitor in the Arduino IDE and place an object in front of each set of sensors. If the IR receiver detects an object, the Serial monitor will display 0’s on the screen. If it does not detect anything, it will display 1’s.

```
/*
* Robotics with the ServoBot - TestIR
* Display 1 if the IR detector does not detect an object, or 0 if it does.
*/

void setup()                             // Built-in initialization block
{
   pinMode(12, INPUT);                   // Right IR LED
   pinMode(11, OUTPUT);                  // Right Receiver

   pinMode(10, INPUT);                   // Left IR LED
   pinMode(9, OUTPUT);                   // LeftReceiver

   Serial.begin(9600);                   // Set data rate to 9600 bps
}

void loop()                              // Main loop auto-repeats
{
   int irRight = irDetect(11, 12, 38000);       // Check for object

   int irLeft = irDetect(9, 10, 38000);  // Check for object

   Serial.print(irLeft);                 // Display 1/0 no detect/detect Left
   Serial.print("   ");                  // space between LeftIR & RightIR
   Serial.println(irRight);              // Display 1/0 no detect/detect Right

   delay(100);                           // 0.1 second delay
}

// IR Object Detection Function

int irDetect(int irLedPin, int irReceiverPin, long frequency)
{
   tone(irLedPin, frequency, 8);         // IRLED 38 kHz for at least 1 ms
   delay(1); // Wait 1 ms
   int ir = digitalRead(irReceiverPin);  // IR receiver -> ir variable
   delay(1);                             // Down time before recheck
   return ir;                            // Return 1 no detect, 0 detect
}
```

Activity 9: Object Detection and Avoidance

Example Sketch – RoamingWithIR

The code below not only includes movement functions, but also contains an irDetect function to that allows the ServoBot to collect data from its IR detectors and use that data to control speed of the servos to avoid objects.

```
/*
* Robotics with the ServoBot - RoamingWithIR
*/

#include <Servo.h>                            // Include servo library

Servo servoLeft;                              // Declare left and right servos
Servo servoRight;

void setup()                                  // Built-in initialization block
{
   pinMode(10, INPUT);                        // Left Receiver
   pinMode(9, OUTPUT);                        // Left IR LED
   pinMode(12, INPUT);                        // Right Receiver
   pinMode(11, OUTPUT);                       // Right IR

   servoLeft.attach(4);                       // Attach left signal to pin 13
   servoRight.attach(3);                      // Attach right signal to pin 12
}

void loop()                                   // Main loop auto-repeats
{
   int irLeft = irDetect(9, 10, 38000);       // Check for object on left
   int irRight = irDetect(11, 12, 38000);     // Check for object on right

   if((irLeft == 0) && (irRight == 0))        // If both sides detect
   {
      backward(1000);                         // Back up 1 second
      turnLeft(800);                          // Turn left about 120 degrees
   }
   else if(irLeft == 0)                       // If only left side detects
   {
      backward(1000);                         // Back up 1 second
      turnRight(400);                         // Turn right about 60 degrees
   }
   else if(irRight == 0)                      // If only right side detects
   {
      backward(1000);                         // Back up 1 second
      turnLeft(400);                          // Turn left about 60 degrees
   }
   else                                       // Otherwise, no IR detected
   {
      forward(20);                            // Forward 1/50 of a second
   }
}

int irDetect(int irLedPin, int irReceiverPin, long frequency)
{
   tone(irLedPin, frequency, 8);            // IRLED 38 kHz for at least 1 ms
   delay(1);                                // Wait 1 ms
   int ir = digitalRead(irReceiverPin);     // IR receiver -> ir variable
   delay(1);                                // Down time before recheck
   return ir;                               // Return 1 no detect, 0 detect
}

void forward(int time)                      // Forward function
{
   servoLeft.writeMicroseconds(1700);       // Left wheel counterclockwise
   servoRight.writeMicroseconds(1300);      // Right wheel clockwise
   delay(time);                             // Maneuver for time ms
}

void turnLeft(int time)                     // Left turn function
{
   servoLeft.writeMicroseconds(1300);       // Left wheel clockwise
   servoRight.writeMicroseconds(1300);      // Right wheel clockwise
   delay(time); // Maneuver for time ms
}

void turnRight(int time)                    // Right turn function
{
   servoLeft.writeMicroseconds(1700);       // Left wheel counterclockwise
   servoRight.writeMicroseconds(1700);      // Right wheel counterclockwise
   delay(time);                             // Maneuver for time ms
}

void backward(int time)                     // Backward function
{
   servoLeft.writeMicroseconds(1300);       // Left wheel clockwise
   servoRight.writeMicroseconds(1700);      // Right wheel counterclockwise
   delay(time);                             // Maneuver for time ms
}
```

Activity 10: High-performance IR Navigation

Example Sketch – FastIrRoaming

The code below is a big change from the previous function examples. It might look confusing at first, but the ServoBot has better performance because the code is checking the IR detectors more often and making smaller movements that do not take as long to complete.

```
/*
* Robotics with the ServoBot - FastIrRoaming
*/

#include <Servo.h>                            // Include servo library

Servo servoLeft;                              // Declare left and right servos
Servo servoRight;

void setup()                                  // Built-in initialization block
{
   pinMode(10, INPUT); pinMode(9, OUTPUT);   // Left IR LED & Receiver
   pinMode(12, INPUT); pinMode(11, OUTPUT);  // Right IR LED & Receiver

   servoLeft.attach(4);                       // Attach left signal to pin 13
   servoRight.attach(3);                      // Attach right signal to pin 12
}

void loop()                                   // Main loop auto-repeats
{

   int irLeft = irDetect(9, 10, 38000);      // Check for object on left
   int irRight = irDetect(11, 12, 38000);    // Check for object on right

   if((irLeft == 0) && (irRight == 0))       // If both sides detect
   {
      maneuver(-200, -200, 20);              // Backward 20 milliseconds
   }
   else if(irLeft == 0)                      // If only left side detects
   {
      maneuver(200, -200, 20);               // Right for 20 ms
   }
   else if(irRight == 0)                     // If only right side detects
   {
      maneuver(-200, 200, 20);               // Left for 20 ms
   }
   else                                       // Otherwise, no IR detects
   {
      maneuver(200, 200, 20);                // Forward 20 ms
   }
}

int irDetect(int irLedPin, int irReceiverPin, long frequency)
{
   tone(irLedPin, frequency, 8);             // IRLED 38 kHz for at least 1 ms
   delay(1);                                  // Wait 1 ms
   int ir = digitalRead(irReceiverPin);      // IR receiver -> ir variable
   delay(1);                                // Down time before recheck
   return ir;                               // Return 1 no detect, 0 detect
}

void maneuver(int speedLeft, int speedRight, int msTime)
{
// speedLeft, speedRight ranges: Backward Linear Stop Linear Forward
//                                 -200   -100.....0...100     200
   servoLeft.writeMicroseconds(1500 + speedLeft);    // Set left servo speed
   servoRight.writeMicroseconds(1500 - speedRight);  // Set right servo speed
   if(msTime==-1)                                    // if msTime = -1
   {
      servoLeft.detach();                            // Stop servo signals
      servoRight.detach();
   }
   delay(msTime);                                    // Delay for msTime
}
```

Activity 11: Detecting Distance

Example Sketch – DisplayBothDistances

The code below very similar to activity 8 on page 35 and tests both sets of IR detectors at the same time. However, instead of two states, 1 for no object, 0 for object, the code generates a range of frequencies from 38,000 Hz to 42,000 Hz in 1000 Hz increments. This gives the code 6 different zones to detect objects. The higher the zone, the farther the object.

After uploading the code to the ServoBot, keep the USB cord connected between the computer and ServoBot. Then open the Serial Monitor in the Arduino IDE and place an object in front of each set of sensors. If the Serial monitor will display 0's on the screen, the IR receiver is detecting an object very close to the ServoBot. The higher the number, the farther the object is from the ServoBot. If detector does not detect anything, it will display a 5.

```
/*
* Robotics with the ServoBot - DisplayBothDistances
* Display left and right IR states in Serial Monitor.
* Distance range is from 0 to 5. Only a small range of several centimeters
* in front of each detector is measureable. Most of it will be 0 (too
* close) or 5 (too far).
*/
void setup()                                 // Built-in initialization block
{
   pinMode(10, INPUT); pinMode(9, OUTPUT);   // Left IR LED & Receiver
   pinMode(12, INPUT); pinMode(11, OUTPUT);  // Right IR LED & Receiver

   Serial.begin(9600);                       // Set data rate to 9600 bps
}

void loop()                                  // Main loop auto-repeats
{
   int irLeft = irDistance(9, 10);           // Measure left distance
   int irRight = irDistance(11, 12);         // Measure right distance

   Serial.print(irLeft);                     // Display left distance
   Serial.print("   ");                      // Display spaces
   Serial.println(irRight);                  // Display right distance

   delay(100);                               // 0.1 second delay
}

// IR distance measurement function

int irDistance(int irLedPin, int irReceivePin)
{
   int distance = 0;
   for(long f = 38000; f <= 42000; f += 1000)
   {
      distance += irDetect(irLedPin, irReceivePin, f);
   }
   return distance;
}

// IR Detection function

int irDetect(int irLedPin, int irReceiverPin, long frequency)
{
   tone(irLedPin, frequency, 8);            // IRLED 38 kHz for at least 1 ms
   delay(1);                                // Wait 1 ms
   int ir = digitalRead(irReceiverPin);     // IR receiver -> ir variable
   delay(1);                                // Down time before recheck
   return ir;                               // Return 1 no detect, 0 detect
}
```

Experiment with different color objects and see how the IR detectors respond. Find a color that works the best and use that color on the object you want the ServoBot to follow. For example, if white works the best, then put a white piece of paper on the back of another ServoBot so the following Bot can track it better.

Activity 12: ServoBot Shadow Vehicle

Example Sketch - FollowingServoBot

The code below takes the irDetect function from the previous activity and uses its output to control servo speed so the ServoBot will maintain the same relative distance to an object, even if that object moves.

```
/*
* Robotics with the ServoBot - FollowingServoBot
* Use proportional control to maintain a fixed distance between
* ServoBot and object in front of it.
*/

#include <Servo.h>                           // Include servo library

Servo servoLeft;                             // Declare left and right servos
Servo servoRight;

const int setpoint = 2;                      // Target distances
const int kpl = -50;                         // Proportional control constants
const int kpr = -50;

void setup()                                 // Built-in initialization block
{
   pinMode(10, INPUT); pinMode(9, OUTPUT);  // Left IR LED & Receiver
   pinMode(3, INPUT); pinMode(2, OUTPUT);   // Right IR LED & Receiver

   servoLeft.attach(4);                      // Attach left signal to pin 13
   servoRight.attach(3);                     // Attach right signal to pin 12
}

void loop()                                  // Main loop auto-repeats
{
   int irLeft = irDistance(9, 10);           // Measure left distance
   int irRight = irDistance(11, 12);         // Measure right distance

// Left and right proportional control calculations
   int driveLeft = (setpoint - irLeft) * kpl;
   int driveRight = (setpoint - irRight) * kpr;

   maneuver(driveLeft, driveRight, 20);      // Drive levels set speeds
}

// IR distance measurement function

int irDistance(int irLedPin, int irReceivePin)
{
   int distance = 0;
   for(long f = 38000; f <= 42000; f += 1000)
   {
      distance += irDetect(irLedPin, irReceivePin, f);
   }
   return distance;
}
```

```
// IR Detection function

int irDetect(int irLedPin, int irReceiverPin, long frequency)
{
   tone(irLedPin, frequency, 8);            // IRLED 38 kHz for at least 1 ms
   delay(1); // Wait 1 ms
   int ir = digitalRead(irReceiverPin);     // IR receiver -> ir variable
   delay(1);                                // Down time before recheck
   return ir;                               // Return 1 no detect, 0 detect
}

void maneuver(int speedLeft, int speedRight, int msTime)
{
// speedLeft, speedRight ranges: Backward Linear Stop Linear Forward
//                                -200    -100....0....100     200
   servoLeft.writeMicroseconds(1500 + speedLeft);    // Set left servo speed
   servoRight.writeMicroseconds(1500 - speedRight);  // Set right servo speed
   if(msTime==-1) // if msTime = -1
   {
      servoLeft.detach(); // Stop servo signals
      servoRight.detach();
   }
   delay(msTime); // Delay for msTime
}
```

Try fine tuning the shadow ServoBot by adjusting the set point and proportionality constants. Use a piece of paper to lead the shadow ServoBot while adjusting the values of kpr and kpl constants from 15 to 100 in the FollowingServoBot code. Notice the responsiveness the ServoBot has when following an object. Also experiment adjusting the setpoint constant value from 0 to 4.

Bonus Activity: If You Want to Follow the Leader

Example Sketch – DisplayBothDistances

The code below is very similar to activity 10 on page 38; however, the maneuver function calls have been modified so the Lead ServoBot travels slower.

```
/*
* Robotics with the ServoBot - SlowerIrRoamingForLeaderBot
* Modification of FastIrRoaming code from activity 10 on page 38. */

#include <Servo.h>                           // Include servo library

Servo servoLeft;                             // Declare left and right servos
Servo servoRight;

void setup()                                 // Built-in initialization block
{
   pinMode(10, INPUT);
   pinMode(9, OUTPUT);                       // Left IR LED & Receiver
   pinMode(3, INPUT);
   pinMode(2, OUTPUT);                       // Right IR LED & Receiver

   servoLeft.attach(13);                     // Attach left signal to pin 13
   servoRight.attach(12);                    // Attach right signal to pin 12
}

void loop()                                  // Main loop auto-repeats
{

   int irLeft = irDetect(9, 10, 38000);      // Check for object on left
   int irRight = irDetect(2, 3, 38000);      // Check for object on right

   if((irLeft == 0) && (irRight == 0))       // If both sides detect
   {
      maneuver(-40, -40, 20);                // Backward 20 milliseconds
   }
   else if(irLeft == 0)                      // If only left side detects
   {
      maneuver(40, -40, 20);                 // Right for 20 ms
   }
   else if(irRight == 0)                     // If only right side detects
   {
      maneuver(-40, 40, 20);                 // Left for 20 ms
   }
   else // Otherwise, no IR detects
   {
      maneuver(40, 40, 20);                  // Forward 20 ms
   }
}

int irDetect(int irLedPin, int irReceiverPin, long frequency)
{
   tone(irLedPin, frequency, 8);            // IRLED 38 kHz for at least 1 ms
   delay(1);                                // Wait 1 ms
   int ir = digitalRead(irReceiverPin);     // IR receiver -> ir variable
   delay(1);                                // Down time before recheck
   return ir;                               // Return 1 no detect, 0 detect
}

void maneuver(int speedLeft, int speedRight, int msTime)
{
// speedLeft, speedRight ranges: Backward Linear Stop Linear Forward
//                                -200    -100....0....100     200
   servoLeft.writeMicroseconds(1500 + speedLeft);    // Set left servo speed
   servoRight.writeMicroseconds(1500 - speedRight);  // Set right servo speed
   if(msTime==-1)                           // if msTime = -1
   {
      servoLeft.detach();                   // Stop servo signals
      servoRight.detach();
   }
   delay(msTime);                           // Delay for msTime
}
```